

AD-A124 145

POLYGON-TO-CHAIN REDUCTIONS AND EXTENSIONS FOR
RELIABILITY EVALUATION OF... (U) CALIFORNIA UNIV BERKELEY
OPERATIONS RESEARCH CENTER R K WOOD OCT 82 ORC-82-12
AFOSR-81-0122 F/G 17/2

1

UNCLASSIFIED

NL

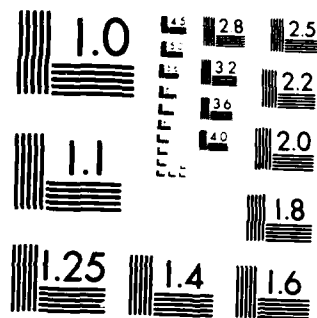
END

DATE

FILED

2-15-1

DTIC



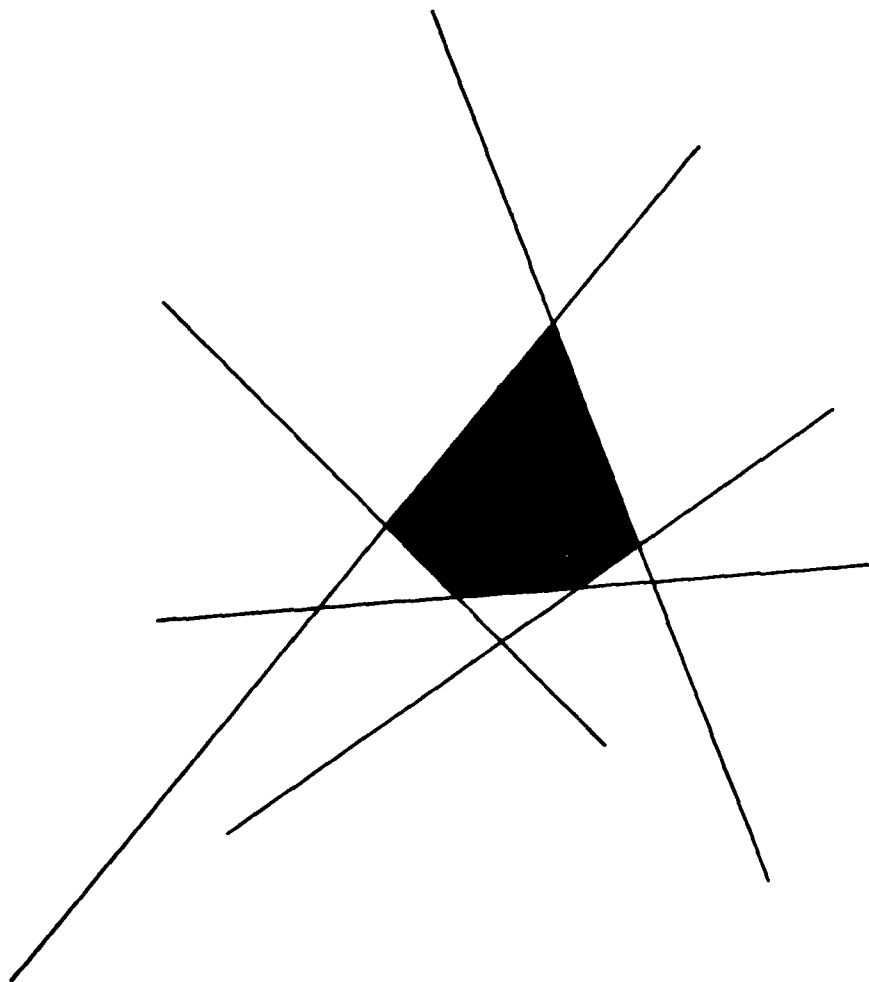
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

ADA 124145

6
ORC 82-12
OCTOBER 1982

POLYGON-TO-CHAIN REDUCTIONS AND EXTENSIONS FOR
RELIABILITY EVALUATION OF UNDIRECTED NETWORKS

by
R. KEVIN WOOD



DTIC FILE COPY

OPERATIONS
RESEARCH
CENTER

DTIC
ELECTE
S FEB 7 1983 D
A

This document has been approved
for public release and sale; its
distribution is unlimited.

UNIVERSITY OF CALIFORNIA • BERKELEY

83 02 07 009

POLYGON-TO-CHAIN REDUCTIONS AND EXTENSIONS
FOR RELIABILITY EVALUATION OF UNDIRECTED NETWORKS

by

R. Kevin Wood
Operations Research Center
University of California, Berkeley

OCTOBER 1982

ORC 82-12

This research has been partially supported by the Air Force Office of Scientific Research (AFSC), USAF, under Grant AFOSR-81-0122 and the National Science Foundation under Grant CEE-8120642 with the University of California. Reproduction in whole or in part is permitted for any purpose of the United States Government.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ORC 82-12	2. GOVT ACCESSION NO. AD-A124145	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) POLYGON-TO-CHAIN REDUCTIONS AND EXTENSIONS FOR RELIABILITY EVALUATION OF UNDIRECTED NETWORKS		5. TYPE OF REPORT & PERIOD COVERED Research Report
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) R. Kevin Wood		8. CONTRACT OR GRANT NUMBER(s) AFOSR-81-0122
9. PERFORMING ORGANIZATION NAME AND ADDRESS Operations Research Center University of California Berkeley, California 94720		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 2304/A5
11. CONTROLLING OFFICE NAME AND ADDRESS United States Air Force Air Force Office of Scientific Research Bolling Air Force Base, D.C. 20332		12. REPORT DATE October 1982
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		13. NUMBER OF PAGES 83
		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES Also supported by the National Science Foundation under Grant CEE-8120642.		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Network Reliability Series-parallel Graphs Efficient Algorithm Graph Reductions Triconnected Decomposition Backtrack Algorithm		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) (SEE ABSTRACT)		

DD FORM 1 JAN 73 1473

EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

To Mom and Dad

DATE	10/10/60
TIME	10:00
LOCATION	1000
DESCRIPTION	1000
REMARKS	1000
SIGNATURE	A
DTIC	
COPY	
INSPECTED	
9	

Acknowledgements

I owe my utmost gratitude, in equal but different ways, to A. Satyanarayana and Richard Barlow. I wish to thank Satya for introducing me to network reliability and for his energetic and optimistic approach to research. His enthusiasm pushed me along when I would rather have gone home and gone to sleep. Professor Barlow's guidance and support made my research possible. I thank him for the foresight and generosity to support me through these last few years even though he could have had a Bayesian instead.

I would like to thank my fellow students Avinash Agrawal, Rubin Johnson and Jane Hagstrom. Their insight helped me time and time again and made the whole doctoral process more rewarding. Also, thanks go to Gloria Partee for her great tables and figures and to the rest of the folks in the O.R. Center for their support.

I particularly want to express my appreciation to Nancy Chapman for her emotional support and for her excellent work in editing this thesis. The pitfalls of "elegant variation" are never more dangerous than in trying to thank so many people. I have learned much.

Finally, I thank the National Science Foundation and the Air Force Office of Scientific Research for supporting this research. May your networks remain reliable.

Abstract

Let $G=(V,E)$ be an undirected graph whose edges may fail, and let G_K denote G with a set $K \subseteq V$ specified. Edge failures are assumed to be statistically independent and to have known probabilities. The K -terminal reliability of G_K , denoted $R(G_K)$, is the probability that all vertices in K are connected by working edges. New reduction techniques are presented which decrease the complexity of computing $R(G_K)$, a task which is NP-hard in general. Reliability-computing algorithms which use these reductions are developed and their complexities are analyzed.

If G is series-parallel, G_K will either be " s - p reducible" or " s - p complex" depending on the configuration of the vertices in K . If G_K is s - p reducible, $R(G_K)$ can be computed in polynomial time by using standard series-parallel reduction techniques which reduce G_K to a single edge whose reliability is trivially evaluated. But if G_K is s - p complex, $R(G_K)$ cannot be evaluated in this way. Until now, only exponential-time algorithms as used on general graphs were known for computing $R(G_K)$ in the s - p complex case. However, by developing a new set of reliability-preserving reductions, we prove that $R(G_K)$ can be computed in polynomial time in the s - p complex case, too.

These new "polygon-to-chain" reductions are of general applicability and always decrease the size of a graph. If a series-parallel graph does not admit a standard reduction, then we prove that it must admit a polygon-to-chain reduction. Combining all types of reductions, an $O(|E|)$ algorithm is presented for computing the reliability of any series-parallel graph irrespective of the vertices in K . The algorithm is extended to make all possible reductions in a graph which is not series-parallel, and results are extended to handle graphs with unreliable vertices.

Suppose $G_K = \dot{G}_K \cup \ddot{G}_K$ such that $\dot{E} \cap \ddot{E} = \emptyset$, $\dot{V} \cap \ddot{V} = \{u, v\}$, $|\dot{E}| \geq 2$ and $|\ddot{E}| \geq 2$. The vertices $\{u, v\}$ are called a separating pair. We prove that any subgraph between a separating pair such as \dot{G}_K may be replaced by a chain of one, two or three edges between u and v such that the reliability of G_K is preserved. This is a generalization of earlier results which showed that for certain configurations of K , a subgraph such as \dot{G}_K could be replaced by a single edge. We show how this reduction can be carried out by computing no more than four K -terminal reliability problems defined on \dot{G}_K . Results are also extended to graphs with unreliable vertices, although in this case the reduction may require the computation of up to six reliability problems defined on \dot{G}_K .

A factoring algorithm for computing network reliability recursively applies the formula $R(G_K) = p_i R(G_K \cdot e_i) + q_i R(G_K - e_i)$ where $G_K \cdot e_i$ is G_K with edge e_i contracted, $G_K - e_i$ is G_K with e_i deleted and $p_i = 1 - q_i$ is the reliability of edge e_i . Various reliability-preserving reductions may be performed after each factoring operation in order to reduce computational complexity. We show that for $2 \leq |K| \leq 5$ or $|V| - 2 \leq |K| \leq |V|$, the complexity of a properly implemented factoring algorithm using standard series, parallel and degree-2 reductions along with the polygon-to-chain reductions will be bounded above by a combinatorial invariant of G called the minimum domination. The factoring algorithm with polygon-to-chain reductions will always perform as well as or better than the algorithm using only standard reductions, and for some networks, it will outperform the simpler algorithm by an exponential factor.

Table of Contents

	page
1. Introduction.....	1
1.1 Graph Theory Fundamentals.....	2
1.2 Formal Model and Problem Definition	3
1.3 Problem Complexity.....	5
1.4 The Factoring Theorem of Network Reliability.....	6
1.5 Thesis Outline and Summary of Results	8
 2. Polygon-to-Chain Reductions and Series-Parallel Graphs.....	 11
2.1 Introduction.....	11
2.2 Preliminaries.....	13
2.3 Polygon-to-Chain Reductions.....	16
2.4 Properties of s - p Complex Graphs.....	24
2.5 An $\$O(E)\$$ Algorithm for s - p Complex Graphs.....	26
2.6 Extension to General Networks	33
2.7 Extension to Networks with Unreliable Vertices.....	34
 3. Triconnected Decomposition for the K-terminal Problem.....	 38
3.1 Introduction.....	38
3.2 Triconnected Decomposition of a Graph.....	39
3.3 Triconnected-Component-to-Chain Reductions	41
3.4 Extension to Unreliable Vertices.....	54
 4. Polygon-to-Chain Reductions and Factoring.....	 58
4.1 Introduction.....	58
4.2 Preliminaries.....	60
4.3 Optimal Factoring Algorithms.....	61
4.4 New Complexity Results for the K-terminal Problem	64
 References.....	 75

Chapter 1

Introduction

Analysis of network reliability is important in computer, communication, power and various other networks. Components of a particular network may be subject to random failure and the network may or may not continue to function after some of its components have failed. We wish, as efficiently as possible, to determine the probability that the network is functional. The purpose of this thesis is to develop new reduction techniques for computing network reliability and to show how computational complexity can be decreased by using these techniques.

The network model which is used in this work may be thought of as a communication network with duplex communication links connecting various transceiving stations. Communication can pass in both directions along a link if the link is working; no communication in either direction is possible if the link has failed. The network is considered functional if a specified set of the transceiving stations is able to communicate.

Although very simple, this model produces difficult theoretical and computational problems. This simplicity may result in the model not being directly applicable to many real-world situations. However, when this is the case, techniques developed here may provide valuable tools for more complex analyses. For example, we will assume that component-failure probabilities are known with certainty. If this is not true, it may be possible to estimate a probability distribution for the reliability of the network by combining techniques of this thesis with numerical integration and/or simulation.

The network model is essentially a probabilistic graph, so, in the next section, a few fundamental graph theoretic notions will be defined. These notions will simplify the formal

definition of the model in the subsequent section.

1.1 Graph Theory Fundamentals

A graph $G=(V,E)$ comprises two finite sets: V is the set of vertices and E is the set of edges. Each edge $e \in E$ corresponds to an unordered pair of vertices, that is, $e=(u,v)$ where $u,v \in V$. The vertices u and v are called the endpoints of edge e . A distinction is sometimes made between a graph and a multigraph. In a graph, no two edges may have both endpoints in common, but this is permissible in a multigraph. For simplicity, we shall allow such parallel edges and still call G a graph. An edge of the form $e=(u,u)$, called a self-loop, is also allowed. A graph with self-loops is often called a pseudo-graph but, again, we use the term "graph" since no confusion results.

If there exists an edge $e=(u,v)$ in G , then u and v are said to be adjacent. Edge e is incident to (with, on) both u and v . The degree of a vertex v , denoted $\deg(v)$, is the number of edges incident on v . Two vertices are connected (or communicate) if there exists a sequence of vertices and edges of the form $u, (u,v_1), v_1, (v_1,v_2), \dots, (v_{l-1},v_l), v_l, (v_l,v), v$. Such a sequence is a path and the path is simple if no vertices are repeated. If the first and last vertices are repeated, the path is a cycle. If only the first and last vertices are repeated, the cycle is simple. A set of vertices $K \subseteq V$ is connected if there exists a path between all pairs of vertices in K . G is said to be connected if V is connected. A tree is a connected graph with no cycles.

A graph $G'=(V',E')$ is a subgraph of $G=(V,E)$ if $V' \subseteq V$ and $E' \subseteq E$. $G-v$ denotes the subgraph of G obtained by deleting vertex v from G , i.e., $G-v=(V-v, E-E_0)$ where $E_0=\{e \in E : v \text{ is an endpoint of } e\}$. If G is connected, but $G-v$ is disconnected, then v is a cutvertex of G . A nonseparable or biconnected graph contains no cutvertices. A block of a graph is a maximal nonseparable subgraph.

Deletion of an edge from a graph can also disconnect that graph. $G-e$ denotes the subgraph obtained from G by deleting edge e from G , i.e., $G-e=(V, E-e)$. If $G-e$ is discon-

nected, then e is a bridge in G .

The reader should consult a standard text such as Harary [1969] for a more basic and comprehensive discussion of graph theory and for any terms not defined here.

1.2 Formal Model and Problem Definition

Let $G=(V,E)$ be a graph whose edges may fail, independently of each other, with known probabilities. Vertices are assumed to be perfectly reliable. The **edge-failure probability** for edge e_i is given by q_i and the **edge reliability** is given by $p_i=1-q_i$. Assume for now that these probabilities are all given for a particular instant of time, τ . Two different interpretations will be provided shortly.

For reliability purposes, a set $K \subseteq V$ must be specified for G . Such vertices will be referred to as the **K-vertices** of G and G_K used to denote the graph G with K specified. Both the terms "graph" and "network" will be used to refer to G_K with or without the associated edge reliabilities since no confusion results. Strictly speaking, however, a network is a graph whose components have additional properties or constraints imposed upon them. Now, the **K-terminal reliability** of G_K , denoted $R(G_K)$, is simply the probability, at time τ , that all K-vertices in G_K are connected by working edges, unless $|K|=1$ in which case $R(G_K) \equiv 1$. (We shall use the shorter phrase " G_K is connected" to mean that the K-vertices in G_K are connected.) For historical reasons, most authors (see Hwang et al. [1981] for a good review and bibliography) have considered the computation of $R(G_K)$ only when $|K|=2$ or $K=V$. However, the case where $2 \leq |K| \leq |V|$ has been handled, in varying degrees of generality, by some authors (Ball [1979, 1980], Buzacott [1980], Johnson [1982], Rosenthal [1977], Satyanarayana [1982], Wood [1980]), and most of the results of this work will be valid for any set K .

Our definition of network reliability can be generalized to handle the case where the vertices in a graph are unreliable, but, unfortunately, this greatly complicates analysis. Nevertheless, many results of this thesis can be extended to handle unreliable vertices and, where possible, extended results will be stated. When vertex failures are considered, it will be assumed

that all K-vertices are completely reliable since any K-vertex failing implies that the whole network has failed.

As previously mentioned, network reliability can have two major interpretations depending on the meaning of the edge-failure probabilities. In the first scenario, consider a network in which the edges fail over time and are not repaired. At any particular instant of time τ , q_i can be interpreted as the probability that edge e_i has failed; perhaps a set of time-to-failure probability distributions $\{F_i(t), i=1, \dots, |E|\}$ is specified and $q_i = F_i(\tau)$. Then $R(G_K)$ is the probability that G_K is still functional at time τ .

Another interpretation can be made for a system in which the components fail, are repaired, fail, are repaired, etc. If these sequences of failure and repair times form independent renewal processes, then the asymptotic failure probabilities are given by $q_i = E[D_i]/(E[D_i] + E[U_i])$ where $E[D_i]$ is the expected length of time edge e_i is down (failed) and $E[U_i]$ is the expected length of time the edge is up (functioning). $R(G_K)$ is then the steady-state probability that the network is functioning (Barlow and Proschan [1975]).

We will limit our discussion in this thesis to graphs which are initially connected and non-separable. Suppose we wish to determine $R(G_K)$ but G is disconnected. If the K-vertices are disconnected then $R(G_K) \equiv 0$ and the situation is distinctly uninteresting. If G is initially disconnected but the K-vertices are connected, then there must exist a connected component or subgraph of G , say G' , which contains all the K-vertices. Thus $R(G_K) = R(G'_K)$ and we may ignore the rest of G . A graph can be tested for connectivity very efficiently but the point is rather moot since in any real-world problem, only a very ill-defined network would be disconnected when working perfectly.

Now, suppose $G=(V,E)$ is connected but separable and let $v \in V$ be any cutvertex of G . G can be partitioned into two connected subgraphs $G^{(1)}=(V_1, E_1)$ and $G^{(2)}=(V_2, E_2)$ such that $V_1 \cup V_2 = V$, $V_1 \cap V_2 = v$, $E_1 \cup E_2 = E$ and $E_1 \cap E_2 = \emptyset$. Also, $E_1 \neq \emptyset$ and $E_2 \neq \emptyset$. Let $K_1 = K \cap V_1$ and $K_2 = K \cap V_2$. If $K_1 - v = \emptyset$ or $K_2 - v = \emptyset$, then the corresponding component is irrelevant and may be disregarded. Otherwise, it is well known that $R(G_K) = R(G^{(1)}_{K_1+v}) R(G^{(2)}_{K_2+v})$. Thus the

reliability of a separable graph can be computed by evaluating the reliabilities of its blocks separately. The blocks of a graph can be identified easily using biconnected decomposition via depth-first-search (Tarjan [1972]), and, for this reason, we henceforth consider only nonseparable graphs.

1.3 Problem Complexity

The K-terminal network reliability problem belongs to the class of NP-hard problems (Ball [1980], Ball and Provan [1981], Rosenthal [1974], Valiant [1977]). This class contains all those problems which are at least as hard as the well-known NP-complete problems such as the satisfiability problem and the traveling salesman problem. In addition, Valiant [1977] and Ball and Provan [1981] have shown that respectively, the two-terminal and all-terminal problems belong to the class of #P complete (number-P complete) problems which implies that they are of similar difficulty to a variety of enumeration problems (counting, not listing) such as counting the number of satisfying assignments of variables in a boolean, conjunctive normal form expression, counting the number of source-to-terminal paths in a network, etc.

In order to get a feel for the inherent difficulty of the network reliability problem, consider the standard test for membership in NP: Given a tentative solution to the problem expressed in decision form, can we check in polynomial time whether or not the solution is valid? For example, the statement for the traveling salesman problem defined on a graph G with given edge lengths is, "Is there a tour of length L or less in the given network?" If a tentative solution, i.e., a tour, is proffered, we can easily check in polynomial time to see if the "tour" is in fact a valid tour and whether or not its length is L or less. Suppose we state the reliability problem as a standard decision problem: "Is $R(G_K) \leq \alpha$ for a given set of edge reliabilities?" No structure (that we know of) like a traveling salesman tour can be submitted as a tentative solution. The only way to check a reliability solution is to go back and compute the reliability of the network once again from the start. Thus it seems that the K-terminal reliability problem is at least one step more difficult than many standard combinatorial optimization problems. Additional weight is given to our argument by the fact that the reliability

approximation problem is NP-hard (Rosenthal [1974]) and by the fact that, unlike many other combinatorial problems, no good heuristic procedures exist for solving the reliability problem.

We are thus dealing with an inherently intractable problem and any general algorithm that we can devise will be of exponential complexity unless someone produces an earth-shattering breakthrough. What can be done? Well, we can attempt to make the algorithms that we do have as efficient as possible, find and exploit topological properties of networks so as to reduce computational complexity and search for special classes of networks which admit efficient, polynomial-time solutions. In this work, contributions are made along all of these lines.

1.4 The Factoring Theorem of Network Reliability

Let $e_i = (u, v)$ be some edge of a graph G_K and let F_i denote the event that e_i is working and \bar{F}_i denote the complementary event. Since $R(G_K)$ is just a probability, the rules of conditional probability can be applied to obtain

$$R(G_K) = p_i R(G_K|F_i) + q_i R(G_K|\bar{F}_i) \quad (1.1)$$

$G_K|F_i$ actually defines a new graph in which u and v are known to be connected. This new induced graph (we do not mean induced subgraph in the standard graph theoretic sense) denoted $G_K * e_i$, is obtained by deleting e_i and merging u and v into a single supervertex $w = u \cup v$. If either u or v is a K -vertex then w is a K' -vertex. More formally, $G_K * e_i$ is defined by

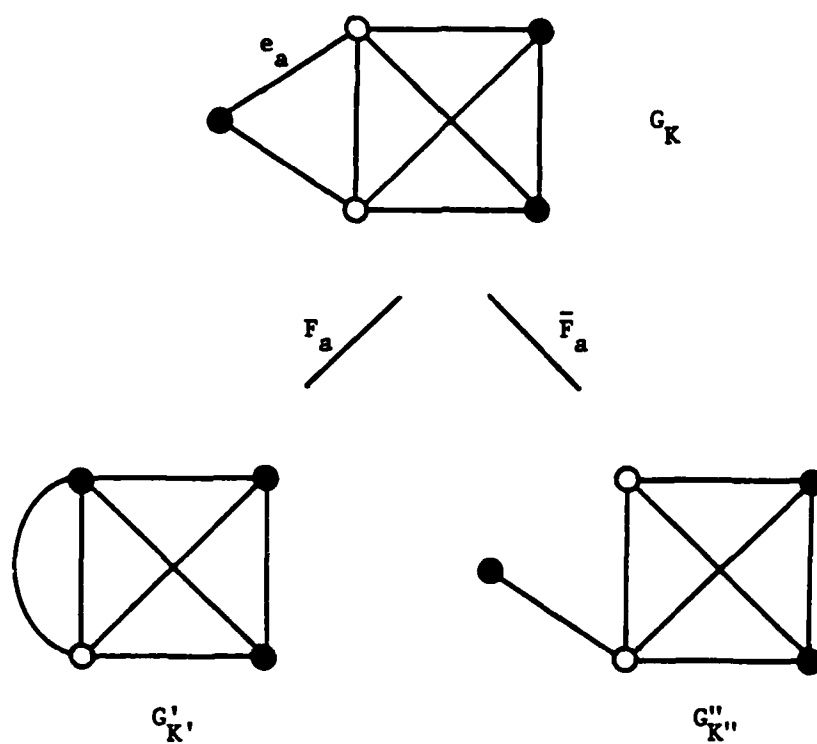
$$\begin{aligned} G * e_i &= (V - u - v + w, E - e_i), \quad w = u \cup v, \\ K' &= \begin{cases} K & \text{if } u, v \notin K \\ K - u - v + w & \text{if } u \in K \text{ or } v \in K \end{cases} \end{aligned} \quad (1.2)$$

Similarly, $G_K|\bar{F}_i$ defines a new graph denoted $G_K - e_i$ where $G - e_i = (V, E - e_i)$. Figure 1.1 illustrates how these two graphs are induced. We can now write Equation 1.1 as

$$R(G_K) = p_i R(G_K * e_i) + q_i R(G_K - e_i) \quad (1.3)$$

That this relationship holds was first shown in Moore and Shannon [1956] and is known as the factoring theorem for network reliability.

This factoring theorem is useful in two ways. First, in Chapters 2 and 3, we will use it as



Induced Graphs Obtained by Factoring on e_a

FIGURE 1.1

a method to derive and prove the validity of some new **reliability-preserving reductions**. Roughly speaking, a reliability-preserving reduction is a technique whereby some subgraph of G_K is replaced by a smaller subgraph to obtain G'_K such that $R(G_K) = \Omega R(G'_K)$ where Ω is a constant obtained from the original subgraph. These reductions are used to compute reliability directly where possible or used in conjunction with other more general methods to reduce computational complexity. The common series and parallel reductions discussed in Chapter 2 are examples of reliability-preserving reductions where $\Omega \equiv 1$.

The factoring theorem is also the basis for a whole class of algorithms for computing network reliability. Moskowitz [1958] was the first to employ the factoring theorem directly as a means of calculating network reliability. Equation 1.3 can be recursively applied to the induced graphs and reliability-preserving reductions made where applicable within the recursion. Eventually the induced graphs are reduced to simple structures, like single edges, for which reliability is trivially computed, or some K-vertices become disconnected, in which case the reliability of the induced graph is zero. In this way, the reliability of any network may be computed, at least in theory. This method of computing network reliability is known as **factoring** and is a special case of pivotal decomposition of a binary coherent system (Barlow and Proschan [1975]). Other methods of network reliability evaluation do exist, including inclusion-exclusion approaches (Satyanarayana and Prabhakar [1978], Satyanarayana [1982]), composition (Buzacott [1980]), simulation (Kumamoto *et al.* [1977], Kumamoto *et al.* [1980]) and boolean-algebra methods (Fratta and Montanari [1973]). (See Hwang *et al.* [1981] for a fairly complete and well-organized listing of works in the different areas.) However, of these general methods, only factoring-based algorithms and their complexity will be discussed in this thesis.

1.5 Thesis Outline and Summary of Results

The remainder of this thesis is outlined in this section. In Chapter 2, we first prove the validity of a new set of reliability-preserving reductions useful for the K-terminal reliability problem. These reductions, called polygon-to-chain reductions, are topologically similar to the well-known parallel reduction but, instead of replacing two edges in parallel with a single edge,

we replace two chains in parallel (a polygon) with a single chain. A chain is basically a simple path all of whose vertices except the first and last have degree 2.

These reductions, along with several standard reductions, are then used to prove that if G has an underlying series-parallel structure, $R(G_K)$ can be computed in polynomial time for any K . The class of graphs which can be analyzed by this method not only includes the well-studied and easily handled two-terminal series-parallel graphs, but also includes a whole class of graphs previously thought to require exponential time to solve. An $O(|E|)$ procedure, Algorithm 2.1, is detailed which either computes $R(G_K)$ if the graph is in the designated class or informs us that the input graph is not a member of the class.

We subsequently modify Algorithm 2.1 in order to apply it to any graph so as to make all possible standard and polygon-to-chain reductions. The modified algorithm can then be used as a subroutine in a factoring algorithm for computing the reliability of any general graph. We conclude Chapter 2 by extending earlier results to the case where vertices not in K are unreliable.

Results of Chapter 3 are direct generalizations of the results of Chapter 2. If we can partition a graph G_K into two parts such that $G_K = \dot{G}_K \cup \ddot{G}_K$, $\dot{E} \cap \ddot{E} = \emptyset$, $\dot{V} \cap \ddot{V} = \{u, v\}$, $|\dot{E}| \geq 2$ and $|\ddot{E}| \geq 2$, then we show that \dot{G}_K or \ddot{G}_K can be replaced by a chain in a reliability-preserving reduction. The pair $\{u, v\}$ is called a separating pair and the reduction described above can be applied in a systematic fashion using triconnected component decomposition which partitions a graph along its separating pairs. Some background information is given on this decomposition and its use in the K -terminal reliability problem. If the vertices of the separating pair are unreliable, our reductions can be validly extended. Details of this extension are presented but are not proven.

Chapter 4 answers the question "What is the minimal complexity of a factoring algorithm which uses standard reductions plus the new polygon-to-chain reductions in its reduction subroutine?" We first specify the general outline of a factoring algorithm and then review some of the earlier work on the complexity of such algorithms. Satyanarayana and Chang [1981] have

used "domination theory" to analyze the complexity of the K-terminal reliability problem using the factoring algorithm along with series and parallel reductions. They relate a graph invariant called domination and denoted $D(G_K)$ to the backtrack search structure produced by the factoring algorithm. They show that the factoring algorithm will be optimal if the number of leaves in its backtrack structure is equal to $D(G_K)$, and show that this will be true if and only if a particular edge-selection strategy is used for factoring. Chang [1981] uses minimum domination, $M(G) = \min_{K:|K|=2} D(G_K)$, to find an optimal factoring algorithm for the all-terminal problem which uses degree-2 and parallel reductions.

We use the same methods to find an optimal edge-selection strategy when $|K|$ is within certain limits. Using a restricted edge-selection strategy, we show that for $2 \leq |K| \leq 5$ or for $|V|-2 \leq |K| \leq |V|$, it is always possible to compute $R(G_K)$ in time which is proportional to $M(G)$. This complexity result is significant since domination may be exponentially larger than minimum domination. It also means that we can compute the most common measure of reliability, two-terminal reliability, in approximately the same amount of time for any two terminals in a given graph. Finally, we remove the restriction on the edge-selection strategy and show that $M(G)$ provides a tight upper bound on algorithmic complexity.

Chapter 2

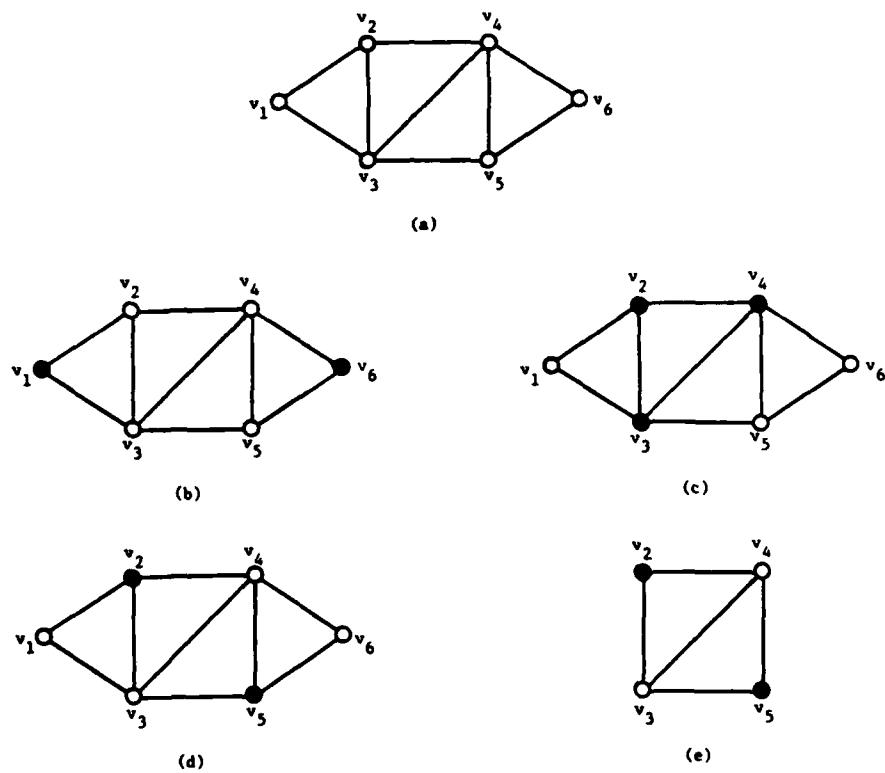
Polygon-to-Chain Reductions and Series-Parallel Graphs

2.1 Introduction

While the network reliability problem is NP-hard for general networks, polynomial-time algorithms do exist for certain network configurations such as "ladders" and "wheels" and for some series-parallel structures such as "two-terminal" series-parallel networks. In this chapter, we show that a class of series-parallel networks, for which only exponentially complex algorithms were previously known (Hänsler *et al.* [1974], Lin *et al.* [1976], Hwang *et al.* [1981]), can be analyzed in polynomial time. In doing this, we introduce a new reliability-preserving reduction of general applicability and produce a linear-time algorithm for computing the reliability of any graph with an underlying series-parallel structure.

In network reliability analysis, three reliability-preserving reductions are well known: the series reduction, the degree-2 reduction (an extension of the series reduction for problems with $|K| > 2$) and the parallel reduction. From the reliability viewpoint, we classify biconnected series-parallel graphs into two broad types, those which are reducible to a single edge using series, parallel and degree-2 reductions, and those which are not. The former type is referred to as *s-p* reducible and the latter, *s-p* complex. For example, the series-parallel graph of Figure 2.1a is *s-p* reducible if $K = \{v_1, v_2\}$, but is *s-p* complex for $K = \{v_1, v_6\}$. Thus, the reducibility of a series-parallel graph, for the purpose of reliability evaluation, depends on the nature of the vertices included in K . A more detailed exposition of this concept appears in section 2.2.

The K -terminal reliability of an *s-p* reducible graph can be computed in polynomial time.



Note: Darkened vertices represent
K-vertices

s - p Reducible and s - p Complex Graphs

FIGURE 2.1

Several methods exist for the solution of the terminal-pair problem for such a graph, i.e., for a two-terminal series-parallel network (Misra [1970], Sharma [1976]), and for $|K| > 2$, direct extensions of the methods can be used. However, researchers have considered computation of the reliability of s - p complex graphs to be as hard as the general problem. The purpose of this chapter is to present an efficient, linear-time algorithm for this problem by introducing a new set of reliability-preserving graph reductions called polygon-to-chain reductions.

In a graph, a chain is an alternating sequence of vertices and edges, starting and ending with vertices such that all internal vertices have degree 2. Two chains with the same end vertices constitute a polygon. In section 2.3, we show that a polygon can be replaced by a chain and that this transformation will yield a reliability-preserving reduction. We discuss the relationship between s - p complex graphs and polygons in section 2.4. Using the polygon-to-chain reductions in conjunction with the three simple reductions mentioned earlier, a polynomial-time procedure is then outlined which will compute the reliability of an s - p complex graph. This procedure is very simple but not necessarily linear, so in section 2.5 we develop, in detail, an efficient algorithm which is shown to operate in $O(|E|)$ time. This algorithm will compute the K -terminal reliability of any graph having an underlying series-parallel structure. Finally, in section 2.6, we discuss how the algorithm can be extended to reduce a nonseries-parallel graph as far as possible so that it could be used as a subroutine in a reliability analysis algorithm for general networks.

2.2 Preliminaries

Simple reductions:

In order to reduce the size of graph G_K and therefore reduce the complexity of computing $R(G_K)$, three well-known simple reductions are often applied. A parallel reduction replaces a pair of edges $e_a = (u, v)$ and $e_b = (u, v)$ with a single edge $e_c = (u, v)$ such that $p_c = 1 - q_a q_b$.

Suppose $e_a = (u, v)$ and $e_b = (v, w)$ such that $u \neq w$, $\deg(v) = 2$ and $v \notin K$. A series reduction replaces e_a and e_b with a single edge $e_c = (u, w)$ such that $p_c = p_a p_b$.

If G'_K is the graph obtained from G_K after a series or parallel reduction, then $R(G_K) = R(G'_K)$. In other words, the K -terminal reliability of G_K remains invariant under series or parallel reductions.

Suppose $e_a = (u, v)$ and $e_b = (v, w)$, such that $u \neq w$, $\deg(v) = 2$ and $\{u, v, w\} \subseteq K$. A **degree-2 reduction** replaces e_a and e_b with a single edge $e_c = (u, w)$ such that $p_c = p_a p_b / (1 - q_a q_b)$ and $R(G_K) = (1 - q_a q_b) R(G'_K - v)$, where G' is the graph obtained from G by replacing e_a and e_b with e_c .

The simple reductions described above are examples of **reliability-preserving reductions** where a subgraph of G_K is replaced by a simpler subgraph to obtain G'_K , where $R(G_K) = \Omega R(G'_K)$ and the multiplicative factor Ω is derived exclusively from the original subgraph. Of course, in the series and parallel reductions $\Omega \equiv 1$ and $K' = K$.

Series-parallel graphs:

The following definition should not be confused with the common definition of a "two-terminal" series-parallel network in which two vertices must remain fixed. No special vertices are distinguished here. In a graph, edges with the same end vertices are **parallel edges**. Two nonparallel edges are **adjacent** if they are incident on a common vertex. Two adjacent edges are **series edges** if their common vertex is of degree 2. Replacing a pair of series (parallel) edges by a single edge is called a **series (parallel) replacement**. A **series-parallel graph** is a graph that can be reduced to a tree by successive series and parallel replacements. Clearly, if a series-parallel graph is nonseparable, then the resulting tree, after making all series and parallel replacements, contains exactly one edge.

We wish to clarify the subtle difference between the term "replacement" used here and the term "reduction" used with respect to simple reductions. By reduction, we mean a reliability-preserving reduction; hence, the term is applicable only to G_K . On the other hand, a replacement is defined on G , irrespective of K . For example, in graph G as shown in Figure 2.1a, series replacements could be made but no reductions are possible in the corresponding G_K with $K = \{v_1, v_6\}$ (Figure 2.1b). Motivated by this distinction, we define an *s-p* reducible graph and

an *s-p* complex graph next.

s-p reducible graphs and s-p complex graphs:

Clearly, if G has no series or parallel edges, then for any K , G_K admits no simple reductions. If G is a series-parallel graph, then a simple reduction might or might not exist in G_K depending upon the vertices of G that are chosen to be in K . For example, consider the series-parallel graph G of Figure 2.1a. The graph G_K , for $K = \{v_2, v_3, v_4\}$ as in Figure 2.1c, can be reduced to a single edge by successive simple reductions. On the other hand, for $K = \{v_1, v_6\}$, G_K has no reductions (Figure 2.1b). A graph G_K is termed *s-p reducible* if it can be reduced to a single edge by successive, simple reductions.

It is possible for a (nonseparable) series-parallel graph to admit one or more simple reductions for a specified K and still not be *s-p* reducible. As an illustration, consider G_K of Figure 2.1d. Two series reductions may be applied to this graph to obtain the graph of Figure 2.1e, but no further simple reductions are possible. A graph G_K is *s-p complex* if G is a series-parallel graph, but G_K is not *s-p* reducible. An *s-p* complex graph may or may not admit some simple reductions.

Chains and polygons:

In a graph, a **chain** χ is an alternating sequence of distinct vertices and edges, $v_1, (v_1, v_2), v_2, (v_2, v_3), v_3, \dots, v_{k-1}, (v_{k-1}, v_k), v_k$, such that the internal vertices, v_2, v_3, \dots, v_{k-1} , are all of degree 2 and the end vertices v_1 and v_k are of degree greater than 2. A chain need not contain any internal vertices, but it must contain at least one edge and the two end vertices. The length of a chain is simply the number of edges it contains. A **subchain** is a connected subset of a chain beginning and ending with a vertex and containing at least one edge. Both the end vertices of a subchain may be of degree 2. The notation χ will also be used for a subchain with the usage differentiated by context.

Suppose χ_1 and χ_2 are two chains of lengths l_1 and l_2 , respectively. If the two chains have common end vertices u and v , then $\chi_1 \cup \chi_2$ is a **polygon** of length $l_1 + l_2$. In other words, a polygon is a cycle with the property that exactly two vertices of the cycle are of degree greater

than 2. While this definition allows two parallel edges to constitute a polygon, we will initially require a polygon to be of length at least 3.

2.3 Polygon-to-Chain Reductions

In this section, a new set of reliability-preserving reductions will be introduced which replaces a polygon with a chain. Consider a graph G_K which does not admit any simple reductions but does contain some polygon Δ . In general, no such Δ need exist, but, if it does exist, then the number of possible configurations is limited.

Property 2.1: Let G_K be a graph which admits no simple reductions. If G_K contains a polygon, it is one of the seven types given in the first column of Table 2.1.

Proof: This follows from the facts that (i) every degree-2 vertex of G_K is a K-vertex, (ii) there can be no more than two K-vertices in a chain, and (iii) the length of any chain in G_K is at most 3. \square

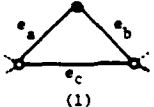
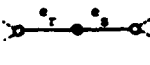
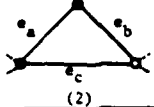
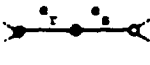
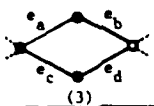
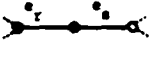
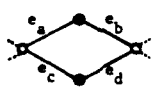

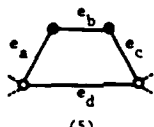
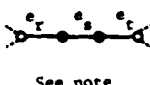
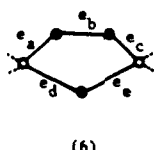
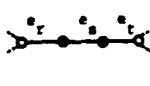
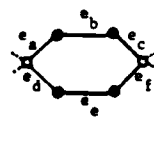
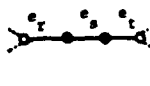
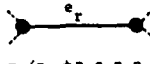
We will use the factoring theorem described in Chapter 1 to prove the validity of our new reduction. For our purposes here, factoring will only be applied to the edges of a single polygon or a chain. To illustrate, consider the graph G_K of Figure 2.2a, which contains a type 1 polygon with 3 edges, e_a , e_b and e_c . Let \underline{F} denote a compound event or "state" such as $F_a \bar{F}_b F_c$ and let \mathbf{F} be the set containing all 2^3 possible states. Also, let $z_i = 1$ if the event F_i occurs in \underline{F} and let $z_i = 0$ if \bar{F}_i occurs. In other words, z_i is an indicator variable which is 1 if e_i works and is 0 if e_i has failed. Equation 1.1 can be extended now to

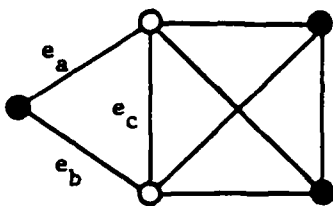
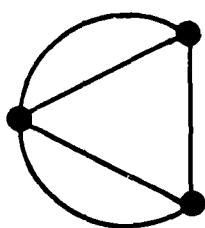
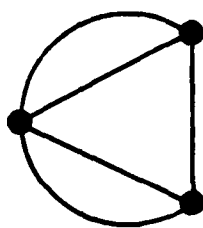
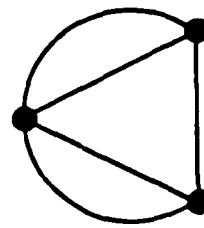
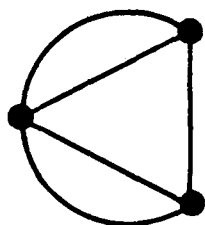
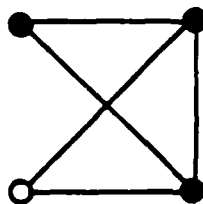
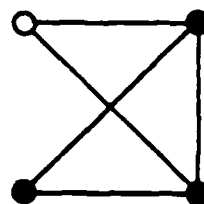
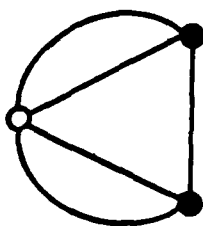
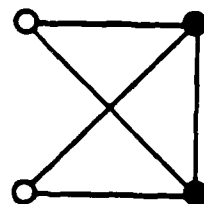
$$R(G_K) = \sum_{\underline{F} \in \mathbf{F}} p_a^{z_a} q_a^{1-z_a} p_b^{z_b} q_b^{1-z_b} p_c^{z_c} q_c^{1-z_c} R(G_K | \underline{F}) \quad (2.1)$$

Figure 2.2b shows the eight graphs induced by the terms of the above equation. Note that four of the induced graphs are identical and that two others are failed, i.e., have zero reliability, since $w \in K$ is disconnected in these graphs. With the above introduction, we are now ready to show that a reliability-preserving polygon-to-chain reduction exists for each of the seven polygons given in Table 2.1.

TABLE 2.1
Polygon-to-Chain Reductions

Note: Darkened vertices represent K-vertices

Polygon Type	Chain Type	Reduction Formulas	New Edge Reliabilities
 (1)		$\alpha = q_a p_b q_c$ $\beta = p_a q_b q_c$ $\delta = p_a p_b p_c \left(1 + \frac{q_a}{p_a} + \frac{q_b}{p_b} + \frac{q_c}{p_c} \right)$	$p_r = \frac{\delta}{\alpha + \delta}$ $p_s = \frac{\delta}{\beta + \delta}$ $\Omega = \frac{(\alpha + \delta)(\beta + \delta)}{\delta}$
 (2)		$\alpha = q_a p_b q_c$ $\beta = p_a q_b q_c$ $\delta = p_a p_b p_c \left(1 + \frac{q_a}{p_a} + \frac{q_b}{p_b} + \frac{q_c}{p_c} \right)$	$p_r = \frac{\delta}{\alpha + \delta}$ $p_s = \frac{\delta}{\beta + \delta}$ $\Omega = \frac{(\alpha + \delta)(\beta + \delta)}{\delta}$
 (3)		$\alpha = p_a q_b q_c p_d + q_a p_b p_c q_d + q_a p_b q_c p_d$ $\beta = p_a q_b p_c q_d$ $\delta = p_a p_b p_c p_d \left(1 + \frac{q_a}{p_a} + \frac{q_b}{p_b} + \frac{q_c}{p_c} + \frac{q_d}{p_d} \right)$	
 (4)		$\alpha = q_a p_b q_c p_d$ $\beta = p_a q_b q_c p_d + q_a p_b p_c q_d$ $\delta = p_a q_b p_c q_d$ $\gamma = p_a p_b p_c p_d \left(1 + \frac{q_a}{p_a} + \frac{q_b}{p_b} + \frac{q_c}{p_c} + \frac{q_d}{p_d} \right)$	
 (5)	$ K > 2$  See note	$\alpha = q_a p_b p_c q_d$ $\beta = p_a q_b p_c q_d$ $\delta = p_a p_b p_c q_d \left(1 + \frac{q_a}{p_a} + \frac{q_b}{p_b} + \frac{q_c}{p_c} + \frac{q_d}{p_d} \right)$	$p_r = \frac{\gamma}{\alpha + \gamma}$ $p_s = \frac{\gamma}{\beta + \gamma}$ $p_t = \frac{\gamma}{\delta + \gamma}$ $\Omega = \frac{(\alpha + \gamma)(\beta + \gamma)(\delta + \gamma)}{\gamma^2}$
 (6)		$\alpha = q_a p_b p_c q_d p_e$ $\beta = p_a q_b p_c (p_d q_e + q_d p_e) + p_b (q_a p_c p_d q_e + p_a q_c q_d p_e)$ $\delta = p_a p_b p_c p_d q_e$ $\gamma = p_a p_b p_c p_d p_e \left(1 + \frac{q_a}{p_a} + \frac{q_b}{p_b} + \frac{q_c}{p_c} + \frac{q_d}{p_d} + \frac{q_e}{p_e} \right)$	
 (7)		$\alpha = q_a p_b p_c q_d p_e p_f$ $\beta = p_a q_b p_c (q_d p_e p_f + p_d q_e p_f + p_d p_e q_f) + p_a p_b p_c p_d (p_d q_e + q_d p_e) + q_a p_b p_c p_d (q_a p_f + p_a q_f)$ $\delta = p_a p_b q_c p_d p_e q_f$ $\gamma = p_a p_b p_c p_d p_e p_f \left(1 + \frac{q_a}{p_a} + \frac{q_b}{p_b} + \frac{q_c}{p_c} + \frac{q_d}{p_d} + \frac{q_e}{p_e} + \frac{q_f}{p_f} \right)$	<p>Note: For $K = 2$, new chain is</p>  $p_r = (p_b + p_a q_b p_c p_d) / \Omega$ $\Omega = p_b + p_a q_b p_c$ Compare Theorem 3.1f

(a) Graph G_K with a Type 1 Polygon $F_a F_b F_c$  $\bar{F}_a F_b F_c$  $F_a \bar{F}_b F_c$  $F_a F_b \bar{F}_c$  $F_a \bar{F}_b \bar{F}_c$  $\bar{F}_a F_b \bar{F}_c$  $\bar{F}_a \bar{F}_b F_c$  $\bar{F}_a \bar{F}_b \bar{F}_c$

(b) Graphs Induced by Factoring on the Edges of the Polygon

FIGURE 2.2

Theorem 1: Suppose G_K contains a type j polygon. Let G'_K denote the graph obtained from G_K by replacing the polygon Δ_j with the chain χ_j having appropriately defined edge probabilities, and let Ω_j be the corresponding multiplication factor, all as shown in Table 2.1. Then, $R(G_K) = \Omega_j R(G'_K)$.

We prove the exactness of reduction 7 only, since reductions 1-6 may be shown in a similar fashion. Figure 2.3 illustrates the use of the theorem on a general graph containing a type 7 polygon and Figures 2.4 and 2.5 are used to illustrate the proof of the theorem. To improve readability in the proof and table, we have dropped the subscript "7" on α , β , δ , γ , and Ω even though, strictly speaking, these parameters are all functions of the reduction type.

Proof of Theorem 1: Let F_i be the event that edge e_i in the polygon is working and let \bar{F}_i be the event that edge e_i has failed. \underline{F} denotes a compound event or state such as $F_a F_b \bar{F}_c F_d \bar{F}_e F_f$ and \mathbf{F} denotes the set of all 2^6 such states. Also, $z_i = 1$ if F_i occurs and $z_i = 0$ if \bar{F}_i occurs. By conditional probability,

$$R(G_K) = \sum_{\underline{F} \in \mathbf{F}} p_a^{z_a} q_a^{1-z_a} \dots p_f^{z_f} q_f^{1-z_f} R(G_K | \underline{F}) \quad (2.2)$$

Only sixteen of the possible sixty-four states are non-failed states where $R(G_K | \underline{F}) \neq 0$. Each non-failed state will induce a new graph with a corresponding set of K-vertices of which there are only four different possibilities. Figure 2.4 gives these four graphs $G_{i,K}$, $i=1,2,3,4$, plus the summed state probabilities in each case, α , β , δ , and γ . Thus, by grouping and eliminating terms, Equation 2.2 is reduced to

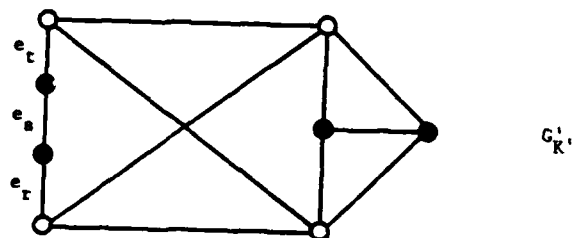
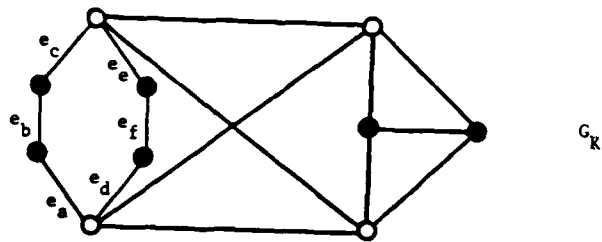
$$R(G_K) = \alpha R(G_{1,K}) + \beta R(G_{2,K}) + \delta R(G_{3,K}) + \gamma R(G_{4,K}) \quad (2.3)$$

Now G'_K is obtained from G_K by replacing the polygon with a chain $u, e_1, v_1, e_2, v_2, e_3, w$ and redefining K as shown in Figure 2.5. Using conditional probabilities again,

$$\begin{aligned} R(G'_K) = & p_r q_s p_t R(G'_K | (F, \bar{F}_3, F_t)) + q_r p_s p_t R(G'_K | (\bar{F}_3, F_s, F_t)) \\ & + p_r p_s q_t R(G'_K | (F, F_s, \bar{F}_t)) + p_r p_s q_t R(G'_K | (F, F_s, F_t)) \end{aligned} \quad (2.4)$$

where only the non-failed states have been written.

The four non-failed states of G'_K induce the same four graphs which the non-failed states



$$\alpha = q_a p_b p_c q_d p_e p_f$$

$$\beta = p_a q_b p_c (q_d p_e p_f + p_d q_e p_f + p_d p_e q_f) \\ + p_a p_b q_c p_f (p_d q_e + q_d p_e) + q_a p_b p_c p_d (q_e p_f + p_e q_f)$$

$$\delta = p_a p_b q_c p_d p_e q_f$$

$$\gamma = p_a p_b p_c p_d p_e p_f \left(1 + \frac{q_a}{p_a} + \frac{q_b}{p_b} + \frac{q_c}{p_c} + \frac{q_d}{p_d} + \frac{q_e}{p_e} + \frac{q_f}{p_f} \right)$$

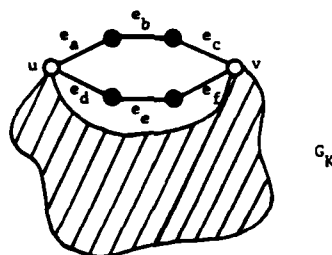
$$p_r = \frac{\gamma}{\alpha + \gamma} \quad p_s = \frac{\gamma}{\beta + \gamma} \quad p_t = \frac{\gamma}{\delta + \gamma}$$

$$\Omega = \frac{(\alpha + \gamma)(\beta + \gamma)(\delta + \gamma)}{\gamma^2}$$

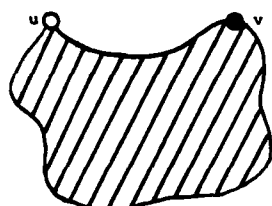
$$R(G_K) = \Omega R(G'_K)$$

Type 7 Polygon-to-Chain Reduction

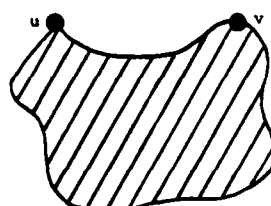
FIGURE 2.3



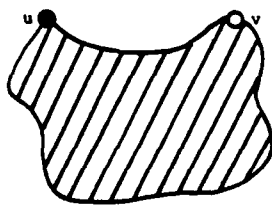
(a) Schematic of a Graph with a Type 7 Polygon



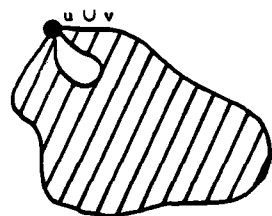
$$\alpha = q_a p_b p_c q_d p_e p_f$$



$$\begin{aligned} \beta &= p_a q_b p_c q_d p_e p_f + p_a q_b p_c p_d q_e p_f \\ &+ p_a p_b q_c q_d p_e p_f + p_a p_b q_c p_d q_e p_f \\ &+ p_a p_b p_c q_d q_e p_f + p_a p_b p_c p_d p_e q_f \\ &+ p_a q_b p_c (q_d p_e p_f + p_d q_e p_f + p_d p_e q_f) \\ &+ p_a p_b q_c p_f (p_d q_e + q_d p_e) \\ &+ q_a p_b p_c p_d (q_e p_f + p_e q_f) \end{aligned}$$



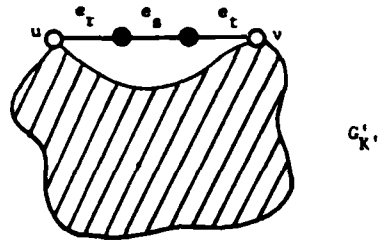
$$\delta = p_a p_b q_c p_d p_e q_f$$



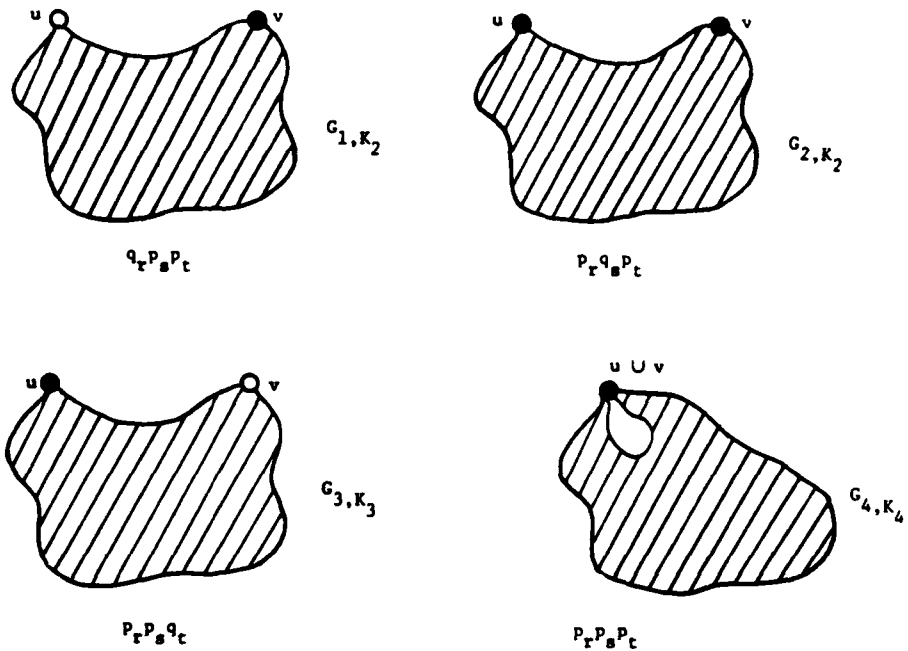
$$\begin{aligned} \gamma &= p_a p_b p_c p_d p_e p_f + q_a p_b p_c p_d p_e p_f \\ &+ p_a q_b p_c p_d p_e p_f + p_a p_b q_c p_d p_e p_f \\ &+ p_a p_b p_c q_d p_e p_f + p_a p_b p_c p_d q_e p_f \\ &+ p_a p_b p_c p_d p_e q_f \\ &+ p_a p_b p_c p_d p_e p_f \left(1 + \frac{q_a}{p_a} + \frac{q_b}{p_b} + \frac{q_c}{p_c} + \frac{q_d}{p_d} + \frac{q_e}{p_e} + \frac{q_f}{p_f} \right) \end{aligned}$$

(b) Non-failed Induced Graphs

FIGURE 2.4



(a) Graph of Fig. 2.4 with Polygon Replaced by Chain



(b) Non-failed Induced Graphs

FIGURE 2.5

of G_K induce. Multiplying Equation 2.4 by a factor Ω , we thus have

$$\begin{aligned} \Omega R(G'_K) &= \Omega p_r q_s p_t R(G_{1,K_1}) + \Omega q_r p_s p_t R(G_{2,K_2}) \\ &+ \Omega p_r p_s q_t R(G_{3,K_3}) + \Omega p_r p_s p_t R(G_{4,K_4}) \end{aligned} \quad (2.5)$$

Equating, term by term, the coefficients in Equations 2.3 and 2.5 gives

$$\begin{aligned} \alpha &= \Omega q_r p_s p_t = \Omega (1-p_r) p_s p_t \\ \beta &= \Omega p_r q_s p_t = \Omega p_r (1-p_s) p_t \\ \delta &= \Omega p_r p_s q_t = \Omega p_r p_s (1-p_t) \\ \gamma &= \Omega p_r p_s p_t \end{aligned}$$

These four equations in the four unknowns Ω , p_r , p_s and p_t may be easily solved to obtain

$$\begin{aligned} p_r &= \frac{\gamma}{\alpha+\gamma} & p_s &= \frac{\gamma}{\beta+\gamma} \\ p_t &= \frac{\gamma}{\delta+\gamma} & \Omega &= \frac{(\alpha+\gamma)(\beta+\gamma)(\delta+\gamma)}{\gamma^2} \end{aligned}$$

which are the values given in Table 2.1 for a type 7 polygon. The reader may verify that when these values are substituted into Equation 2.4, we obtain

$$\begin{aligned} \Omega R(G'_K) &= \alpha R(G_{1,K_1}) + \beta R(G_{2,K_2}) + \delta R(G_{3,K_3}) + \gamma R(G_{4,K_4}) \\ &= R(G_K) \quad \square \end{aligned}$$

Theorem 1 can be extended to give a result which can be useful for computing the reliability of a general graph. In a nonseparable graph, a pair of vertices (u, v) is a **separating pair** if $G = \dot{G} \cup \ddot{G}$ where $|\dot{E}| \geq 2$, $|\ddot{E}| \geq 2$, $\dot{E} \cap \ddot{E} = \emptyset$ and $\dot{V} \cap \ddot{V} = \{u, v\}$. For example, vertices u and v in Figure 2.5 are a separating pair. Using the same conditioning arguments as in the proof of Theorem 1, it can be shown that any subgraph between a separating pair can be replaced by a chain of 1, 2, or 3 edges to yield a reliability-preserving reduction. For two special cases, it has been shown that a subgraph between a separating pair can be replaced by a single edge (Rosenthal [1974]). The first case arises when the subgraph including the separating pair has no K -vertices, and the second case arises when the separating pair belongs to K . The fact that a chain can always be used to replace any subgraph, irrespective of the K -vertices, greatly increases the generality of any algorithm which uses this reduction. In chapter 3 we will discuss in detail how this extended reduction can be used and how the edge reliabilities for the derived chains can be computed in an efficient manner.

2.4 Properties of s - p Complex Graphs

In this section we set down some properties of series-parallel and, in particular, s - p complex graphs. We prove that s - p complex graphs must admit a polygon-to-chain reduction if all simple reductions have first been performed. Using this fact, we then outline a simple polynomial-time procedure for computing the reliability of such graphs.

The following property is a simple extension of the definition of a series-parallel graph.

Property 2.2: Let G' be the graph obtained from G by applying one or more of the following operations:

- A series replacement;
- A parallel replacement;
- An inverse series replacement (replace an edge by two edges in series);
- An inverse parallel replacement (replace an edge by two edges in parallel).

Then, G' is a series-parallel graph if and only if G is series-parallel.

Proof of Property 2.2 may be found in Duffin [1965]. The next property states that the series-parallel structure of a graph is not altered by simple or polygon-to-chain reductions.

Property 2.3: Let G'_K be the graph obtained from G_K by applying a simple reduction or a polygon-to-chain reduction on G_K . Then, G' is a series-parallel graph if and only if G is series-parallel.

Proof: A series or degree-2 reduction implements a series replacement, and a parallel reduction implements a parallel replacement on G . A polygon-to-chain reduction can be considered to implement a sequence of replacements on G : first one or more series replacements, then a parallel replacement, and finally one or more inverse series replacements. Hence, by application of Property 2.2 one or more times, G' is series-parallel if and only if G is series-parallel. \square

An important implication of Property 2.3 is that, if G_K is s - p complex, then application of a simple reduction to G_K results in a graph which again is s - p complex. On the other hand, a polygon-to-chain reduction on G_K results in a graph which is either s - p complex or s - p reducible.

ble. By next proving that every s - p complex graph G_K admits a simple reduction or a polygon-to-chain reduction, it will be possible to show that $R(G_K)$ can be computed in polynomial time for such graphs.

Property 2.4: Let G_K be an s - p complex graph. Then, G_K must admit either a simple reduction or one of the seven types of polygon-to-chain reductions given in Table 2.1.

Proof: If G_K admits a simple reduction, then we are done. If G_K has no simple reductions, then by Property 2.1, any polygon of G_K must be one of the seven types given in Table 2.1. Hence, we need only show that G contains a polygon. Let G' be the graph obtained by replacing all chains in G with single edges. If G' contains a pair of parallel edges, then the two chains in G corresponding to this pair of edges constitute a polygon. We argue that G' must contain a pair of parallel edges. If G' has no parallel edges, no simple reductions are possible in G' since all vertices in G' have degree greater than 2. Thus, G' and hence G are not series-parallel graphs, which is a contradiction. \square

One simple procedure for computing $R(G_K)$ can now be outlined as follows: (1) Make all simple reductions; (2) find a polygon and make the corresponding reduction; and (3) repeat steps 1 and 2 until G_K is reduced to a single edge. If G_K is originally s - p complex, then Properties 2.3 and 2.4 guarantee that the above procedure eventually reduces G_K to a single edge. The actual reliability is calculated by initializing $M \leftarrow 1$, letting $M \leftarrow M\Omega_j$ whenever a polygon-to-chain reduction of type j is done, and letting $M \leftarrow M(1 - q_a q_b)$ whenever a degree-2 reduction is done on some edges e_a and e_b . At the end of the algorithm with a single remaining edge e_i , the reliability of the original graph is given by $R(G_K) = Mp_i$.

The total number of parallel and polygon-to-chain reductions executed by this procedure, before the graph is reduced to a single edge, is exactly $|E| - |V| + 1$. This is because the number of fundamental cycles in a connected graph is $|E| - |V| + 1$, and a parallel or polygon-to-chain reduction deletes exactly one such cycle (Deo [1974]). The complexity of steps (1) and (2) above can be linear in the size of G , and thus, the running time of the whole procedure is at best quadratic in the size of G . In order to develop a linear-time algorithm, we have found it

necessary to move the parallel reduction from the domain of simple reductions to the domain of polygon-to-chain reductions. Indeed, a parallel reduction is a trivial case of a polygon-to-chain reduction with a multiplier $\Omega=1$. We will henceforth consider two parallel edges to be the type 8 polygon and the parallel reduction to be the type 8 polygon-to-chain reduction.

2.5 An $O(|E|)$ Algorithm for s - p Complex Graphs

The objective in this section is to develop an efficient, linear-time algorithm for computing the reliability of an s - p complex graph. This algorithm should also compute the reliability of an s - p reducible graph as a special case and tell us if the graph is not series-parallel. All results needed to present this algorithm have been established; however, some additional notation and definitions must be given.

If u and v are the end vertices of a chain χ , then u and v are said to be **chain-adjacent**. When it is necessary to distinguish these vertices, we will use the notation $\chi(u, v)$. A subchain with end vertices u and v will also be denoted $\chi(u, v)$, but in this case u and v cannot be said to be chain-adjacent. If Δ is a polygon formed by two chains $\chi_1(u, v)$ and $\chi_2(u, v)$, then we use the notation $\Delta(u, v) = \chi_1(u, v) \cup \chi_2(u, v)$. The algorithm is presented next, followed by a short discussion and then a proof of its validity and linear complexity.

Algorithm 2.1

MAIN

Input: A nonseparable graph G with vertex set V , $|V| \geq 2$, edge set E , $|E| \geq 2$, and set $K \subseteq V$, $|K| \geq 2$. Edge reliabilities p_e for each edge $e \in E$.

Output: $R(G_K)$ if G is series-parallel or a message that G is not series-parallel.

Variables: G_K and all vertex "marks" are represented by global data structures and variables. All other variables are local.

- (1) (Initialize) $M \leftarrow 1$.
- (2) (Initialize list) Construct list, $T \leftarrow \{v \mid v \in V \text{ and } \deg(v) > 2\}$ marking all such v "onlist."
- (3) (Perform all series and degree-2 reductions)
 - (a) For each vertex $v \in V$ such that $\deg(v) = 2$ and $v \notin K$, SERIESREDUCE(v).
 - (b) For each vertex $v \in V$ such that $\deg(v) = 2$ and $v \in K$, and while $|E| > 2$, DEGREE2REDUCE(v, M).

- (4) If T is empty then
 - (a) (G may be reduced to two parallel edges) If $|E|=2$ then Print("R(G_K)=" $M(1-q_a q_b)$) and STOP.
 - (b) (Otherwise G has not been completely reduced) Print(" G is not series-parallel") and STOP.
- (5) (T is not empty) Remove any v from T and mark v "offlist."
- (6) (v may have been involved in a reduction since it was put on the list) If $\deg(v)=2$ or v is marked "deleted" then go to (4).
- (7) (Begin search or continue search for a polygon with one endpoint at v) Search chains emanating from v until one of two cases occurs:
 - (a) (v is found to be chain-adjacent to 3 distinct vertices) 3 chains $\chi(v, u_1)$, $\chi(v, u_2)$, $\chi(v, u_3)$ are found such that $u_1 \neq u_2 \neq u_3 \neq u_1$. In this case go to (4).
 - or
 - (b) A polygon $\Delta(v, w) = \chi_1(v, w) \cup \chi_2(v, w)$ is found.
- (8) (A polygon has been found, make the polygon-to-chain reduction) $\text{POLYREDUCE}(\Delta(v, w), \chi(v, w), M)$.
- (9) If $|E|=|V|$ then (G has been reduced to a single cycle)
 - (a) $T \leftarrow \emptyset$.
 - (b) Go to (3).
- (10) (G has not been reduced to a single cycle. Four cases can arise depending on the new degrees of v and w .)
 - (a) If $\deg(v) > 2$ and $\deg(w) > 2$ then go to (7).
 - (b) If $\deg(v) > 2$ and $\deg(w) = 2$ then
 - (i) $\text{CHAINREDUCE}(\chi(v, w), \chi(v, y), M)$.
 - (ii) If y is "offlist" then mark y "onlist" and put y on T .
 - (iii) Go to (7).
 - (c) If $\deg(v) = 2$ and $\deg(w) > 2$ then
 - (i) $\text{CHAINREDUCE}(\chi(v, w), \chi(x, w), M)$.
 - (ii) If x is "offlist" then mark x "onlist" and put x on T .
 - (iii) If w is "offlist" then mark w "onlist" and put w on T .
 - (iv) (Since T cannot be empty) Go to (5).
 - (d) (Otherwise $\deg(v) = 2$ and $\deg(w) = 2$)
 - (i) $\text{CHAINREDUCE}(\chi(v, w), \chi(x, y), M)$.
 - (ii) If x is "offlist" then mark x "onlist" and put x on T .
 - (iii) If y is "offlist" then mark y "onlist" and put y on T .
 - (iv) (Since T cannot be empty) Go to (5).

End of MAIN.

SERIESREDUCE(v)

Input: A vertex v such that $v \in K$ and $\deg(v) = 2$.

(This routine reduces G_K by making a series reduction on the two edges incident on v .)

- (1) Let x and y be the vertices adjacent to v and let $e_a = (v, x)$ and $e_b = (v, y)$.
- (2) (Carry out the reduction)
 - (a) Delete edges e_a and e_b from G_K .
 - (b) Mark v "deleted."
 - (c) Add new edge $e_c = (x, y)$ to G_K .
 - (d) $p_c \leftarrow p_a p_b$
- (3) Return.

End of SERIESREDUCE

DEGREE2REDUCE(v, M)

Input: A vertex v such that $v \in K$ and $\deg(v) = 2$. Multiplier M .

Output: Revised value of multiplier M .

(This routine reduces G_K by performing a degree-2 reduction at v if v is adjacent to two K -vertices.)

- (1) Let x and y be the vertices adjacent to v and let $e_a = (v, x)$ and $e_b = (v, y)$.
- (2) If $x, y \in K$ then
 - (a) Delete edges e_a and e_b from G_K .
 - (b) Mark v "deleted."
 - (c) Add new edge $e_c = (x, y)$ to G_K .
 - (d) $p_c \leftarrow (p_a p_b) / (1 - q_a q_b)$
 - (e) $M \leftarrow M(1 - q_a q_b)$
- (3) Return.

End of DEGREE2REDUCE

CHAINREDUCE($\chi(v, w), \chi(s, t), M$)

Input: A subchain $\chi(v, w)$ obtained in a polygon-to-chain reduction of a polygon $\Delta(v, w)$. A multiplier M .

Output: A completely reduced chain $\chi(s, t)$ obtained from the chain containing the subchain $\chi(v, w)$. New multiplier M .

(This routine finds the chain containing the subchain $\chi(v, w)$ and makes any series and degree-2 reductions to this chain in G_K .)

- (1) If $\deg(v) > 2$ and $\deg(w) = 2$ then
 - (a) $s \leftarrow v$
 - (b) Search from w away from v to find the first vertex t such that $\deg(t) > 2$.
- (2) If $\deg(v) = 2$ and $\deg(w) > 2$ then
 - (a) $t \leftarrow w$
 - (b) Search from v away from w to find the first vertex s such that $\deg(s) > 2$.

- (3) If $\deg(v)=2$ and $\deg(w)=2$ then
 - (a) Search from w away from v to find the first vertex t such that $\deg(t)>2$.
 - (b) Search from v away from w to find the first vertex s such that $\deg(s)>2$.
- (4) Define the new chain $\chi(s, t)$ which is a superset of the subchain $\chi(v, w)$.
- (5) (Make any possible series and degree-2 reductions on $\chi(s, t)$ in G_K)
 - (a) For each vertex $u \in \chi(s, t)$ such that $\deg(u)=2$ and $u \notin K$, SERIESREDUCE(v).
 - (b) For each vertex $u \in \chi(s, t)$ such that $\deg(u)=2$ and $u \in K$, DEGREE2REDUCE(u, M).
- (6) Return.

End of CHAINREDUCE

POLYREDUCE($\Delta(v, w), \chi(v, w), M$)

Input: Polygon $\Delta(v, w)$ and multiplier M .

Output: Chain or subchain $\chi(v, w)$ resulting from polygon-to-chain reduction of $\Delta(v, w)$ and new multiplier M .

(This routine reduces G_K by making the polygon-to-chain reduction on $\Delta(v, w)$ in G_K . It returns the new multiplier M and the chain or subchain $\chi(v, w)$ resulting from the reduction.)

- (1) Determine which of the 8 types of polygons $\Delta(v, w)$ is, say type j .
- (2) If $j=8$ then ($\Delta(v, w)$ is two edges in parallel)
 - (a) Let e_a and e_b be the two edges forming $\Delta(v, w)$.
 - (b) $p_a \leftarrow 1 - q_a q_b$
 - (c) Delete e_b from G_K .
 - (d) Let $\chi(v, w)$ be e_a .
- (2) If $j \leq 7$ then (Apply Theorem 1)
 - (a) Update G_K by replacing $\Delta(v, w)$ with appropriate chain $\chi(v, w)$ as given in Table 2.1.
 - (b) Compute edge probabilities for edges in $\chi(v, w)$ using the appropriate formulas in Table 2.1.
 - (c) Compute Ω_j from Table 2.1.
 - (d) $M \leftarrow M \Omega_j$
- (4) Return.

End of POLYREDUCE

Simplicity and clarity dictate that the algorithm be presented in a form which is not completely structured. The primary departure from a structured program occurs at Step (7) of MAIN where chains emanating from a vertex v are searched. Here the algorithm searches until it finds that v is chain-adjacent to three distinct vertices or until it finds a polygon. If three chain-adjacent vertices are found, the next vertex from the list T is checked as long as T

is not empty. If a polygon is found, then it is reduced by a call to POLYREDUCE. If $\deg(v)$ remains greater than 2 after the reduction, the algorithm returns to Step (7) and continues the search; no chains searched previously from v need be searched again.

One expediency has been the reduction of G to two parallel edges instead of to a single edge. This device simplifies the program and allows us to avoid performing a polygon-to-chain reduction on something that is not strictly a polygon by our definition, because both end vertices are of degree 2. One final comment is that the list T could be any sort of simple linked list, since the order in which the vertices are inserted and removed is unimportant. A stack would be a convenient implementation.

The correctness of the algorithm is not hard to show. Arguments similar to those presented here may be found in Valdes *et al.* [1981], where the problem is the recognition of two-terminal series-parallel directed graphs. Suppose first that G consists of a single cycle. The series and degree-2 reductions at Step (3) (all steps are in MAIN) will reduce G_K to two edges in parallel and T will be empty. The algorithm therefore gives $R(G_K)$ at Step (4.a).

Next, suppose that G does not consist of a single cycle, in which case T will not be empty and an initial search for a polygon will begin at Step (7). Since all initial series and degree-2 reductions were performed at Step (3), by Property 2.4 any polygon found must be one of the eight specified types. If a polygon is found and then reduced at Step (8), the resulting chain may in fact be a subchain. If this happens, some new series and degree-2 reductions may be admitted on the chain containing the subchain. These reductions are made at Step (10.b), (10.c), or (10.d). Thus, every time Step (7) is entered, the graph admits no series or degree-2 reductions, and any polygon found will be one of the eight given types.

Vertices are continually removed from the list T and replaced, at most two at a time, only when reductions are made. Since only a finite number of reductions can be made, T must eventually become empty. If $|E|=2$ at that point, then $R(G_K)$ is correctly given at Step (4.a) since only reliability-preserving series, degree-2, and polygon-to-chain reductions are ever performed. Property 2.4 proves that the original graph must have been series-parallel. Suppose

instead that $|E| > 2$ when T becomes empty. In this case, every vertex v with $\deg(v) > 2$ is chain-adjacent to at least three distinct vertices. This is true since (i) every vertex v with $\deg(v) > 2$ is initially put on the list and its chain-adjacent vertices checked at Step (7), and (ii) whenever the chain-adjacency of a vertex or vertices is altered (this can occur to at most two vertices at a time) at Step (8), this vertex or vertices are returned to the list if not already there. The following property proves that a graph with the given chain-adjacency structure is not series-parallel.

Property 2.5: Let G be a nonseparable graph such that all vertices v with $\deg(v) > 2$ are chain-adjacent to at least three distinct vertices. Then, G is not a series-parallel graph.

Proof: Let G' be the graph obtained from G by first replacing all chains with single edges in a sequence of series replacements and then removing any parallel edges in a sequence of parallel replacements. By Property 2.2, G is series-parallel if and only if G' is series-parallel. Now, every vertex $v \in V'$ has $\deg(v) > 2$ and there are no parallel edges in E' . Thus, G' admits no series or parallel replacements and cannot be series-parallel. Therefore G cannot be series-parallel. \square

This proves that if the algorithm terminates with $|E| > 2$, the reduced graph is not series-parallel, and Property 2.3 proves that the original graph could not have been series-parallel either. Thus, the validity of the algorithm is established. We now turn our attention to the algorithm's computational complexity.

In order to show that the algorithm is linear in the size of G , we must provide more details of its implementation. We use a multi-linked adjacency list form to represent the given graph G . In this representation, for each vertex a doubly-linked list of adjacent vertices corresponding to incident edges is kept together with the associated edge probabilities. Every edge is represented twice since we are dealing with an undirected graph, and additional links are kept between both representations of each edge. Such an adjacency list can be initialized in $O(|V| + |E|)$ time for any graph. However, our assumption that the input graph is nonseparable and contains at least two edges implies that $|V| \leq |E|$, and therefore, the complexity is simply

$O(|E|)$. For the same reason, the complexity of the whole algorithm will be $O(|E|)$.

Using the graph representation described above, it is obvious that any series or degree-2 reduction can be carried out in constant time. Since POLYREDUCE only needs to check for eight different types of polygons, all of limited size, any polygon-to-chain reduction can also be carried out in constant time. Also, none of the reductions ever requires the use of more vertices or edges after the reduction than before. This means that if any new edges or vertices must be defined, old ones can be reused and the size of the graph representation is never increased.

Now, Steps (2) and (3) in MAIN can be performed in $O(|V|)$ time; therefore, we need only consider the central portion of MAIN, Steps (4) through (10). Each time chains emanating from the current vertex v are checked at Step (4), the maximum amount of work which can be performed is some constant amount, i.e., the amount needed to find three chains with distinct end vertices u_1, u_2 and u_3 , plus some amount of work proportional to the number of polygon-to-chain reductions made from v . Initially, at most all the vertices can be on the list, and after every polygon-to-chain reduction, at most two vertices can be returned to the list. An upper bound on the number of vertices which can ever be checked is therefore $|V| + 2(|E| - |V|) = 2|E| - |V|$, since at most $|E| - |V|$ polygon-to-chain reductions can ever be performed by POLYREDUCE. For some constant C_1 , the total amount of work required until T becomes empty will thus be bounded by $C_1(2|E| - |V|)$ plus the amount of work required to make all polygon-to-chain reductions using POLYREDUCE and the subchain reductions using CHAINREDUCE.

We have already shown that the amount of work required by a polygon-to-chain reduction in POLYREDUCE is bounded by a constant. However, after a call to POLYREDUCE which reduces $\Delta(v, w)$ to $\chi(v, w)$, a call to CHAINREDUCE is necessary if $\deg(v) = 2$ or $\deg(w) = 2$. The chain $\chi(s, t)$ containing the subchain $\chi(v, w)$ must be identified and then reduced, if possible, with series and degree-2 reductions. This can be accomplished in constant time also, since the length of any chain $\chi(s, t)$ is at most 9. This worst case could occur if $\deg(v) = \deg(w) = 2$

after the polygon-to-chain reduction of $\Delta(v, w)$ to $\chi(v, w)$, and the subchains $\chi(s, v)$, $\chi(w, t)$ and $\chi(v, w)$, which were proper chains before the reduction, are at their maximum possible lengths of 3. It therefore follows that the amount of work associated with a polygon-to-chain reduction, a call to POLYREDUCE and a possible call to CHAINREDUCE, is bounded by some constant, say C_2 , and that the amount of work associated with all polygon-to-chain reductions is bounded by $C_2(|E| - |V|)$. We can now see that the amount of work required until T is empty is bounded by $C_1(2|E| - |V|) + C_2(|E| - |V|)$. The only work which is unaccounted for comes from the final series and degree-2 reductions which may be necessary if G is reduced to a single cycle, but again, this is an $O(|V|)$ operation. Thus, we have proven the following theorem:

Theorem 2. Let G be a nonseparable series-parallel graph. Then, for any K , $R(G_K)$ can be computed in $O(|E|)$ time.

2.6 Extension to General Networks

The algorithm of section 5 can be extended to make all possible simple and polygon-to-chain reductions in a nonseries-parallel graph. In this way, the extended algorithm can be used as a subroutine in a more general network reliability algorithm for computing $R(G_K)$ when G is not series-parallel. The complexity of computing $R(G_K)$ can often be reduced to some degree by this device.

Suppose the reduction algorithm of section 5 starts with a nonseries-parallel graph G . After termination of the algorithm, G_K may or may not have been partially reduced. From the proof of Property 2.5, the only possible remaining reductions are polygon-to-chain reductions. Each such polygon-to-chain reduction would correspond to a parallel edge replacement used to obtain the graph G' of that proof. Therefore, G_K can be totally reduced by first applying the algorithm and then finding and reducing any remaining polygons, which can easily be done by searching all chains emanating from all vertices v with $\deg(v) > 2$.

To extend the reduction algorithm as described, Step (4) in MAIN may be replaced by

the following, Step(4'):

Algorithm extension, changes to MAIN

(4') If T is empty then

- (a) (G may be reduced to two parallel edges, e_a and e_b) If $|E|=2$ then
 - (i) $R(G_K) \leftarrow M(1 - q_a q_b)$.
 - (ii) Return to general algorithm with $R(G_K)$.
- (b) (Otherwise G has not been completely reduced) For each $v \in V$ such that $\deg(v) > 2$, search all chains emanating from v calling $\text{POLYREDUCE}(\Delta(v, w), \chi(v, w), M)$ whenever a polygon $\Delta(v, w) = \chi_1(v, w) \cup \chi_2(v, w)$ is found.
- (c) Return to general algorithm with reduced G_K and with M .

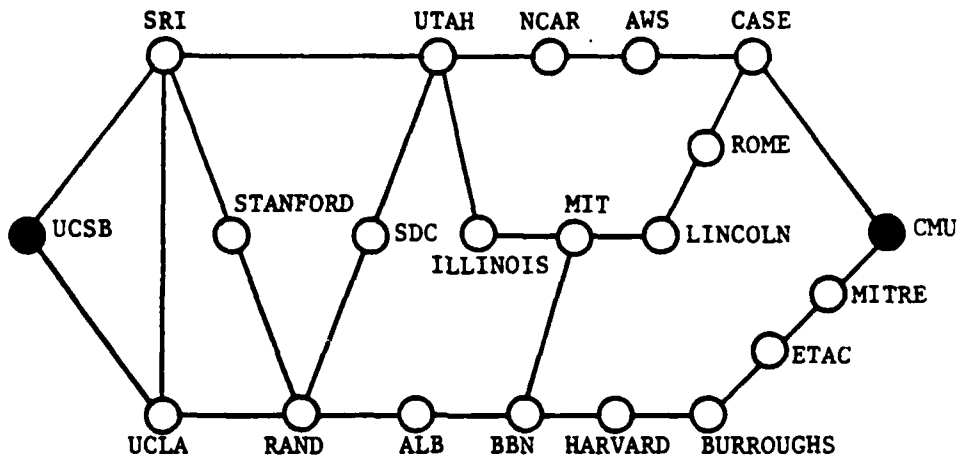
In the worst case at Step (4'.b), each chain and thus each edge must be searched twice. Therefore, the added computation is $O(|E|)$ and the algorithm with the extension remains $O(|E|)$.

To illustrate the usefulness of the extended algorithm for a general graph, let us consider the ARPA computer network configuration as shown in Figure 2.6a (Fratta and Montanari [1973]). Suppose we are interested in the terminal-pair reliability between UCSB and CMU. Application of the extended algorithm yields a reduced network as shown in Figure 2.6b with redefined edge reliabilities and an associated multiplier. The original reliability problem is now equivalent to computing the terminal-pair reliability between RAND and CMU in the reduced network. In linear time the size of the network has been reduced considerably and, because computing the reliability of a general network is exponential in its size, a significant computational advantage should be gained.

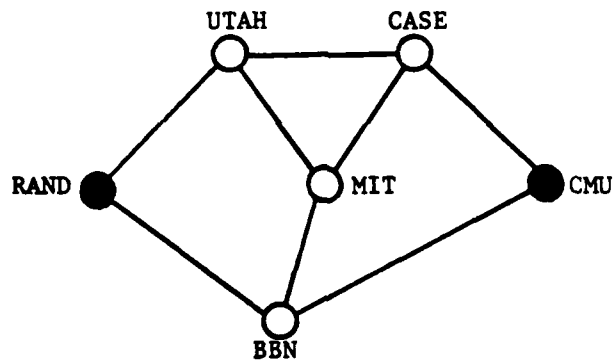
2.7 Extension to Networks with Unreliable Vertices

For the reasons mentioned in Chapter 1, networks with unreliable vertices are not being covered in any depth in this paper. However, the reductions and algorithms of this chapter can be easily extended to handle unreliable vertices and therefore the necessary details are included here.

Only two basic changes need be made: The series reduction must be modified to include the failure probability of the degree-2 vertex and polygon-to-chain reductions must be modified



(a) ARPA Computer Network



(b) Reduced Network

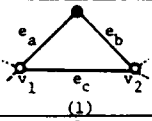
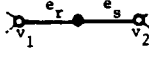
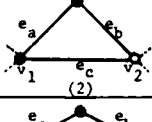
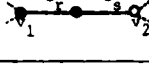
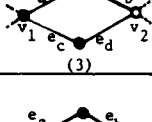
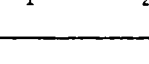
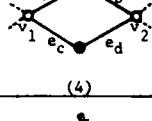

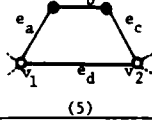

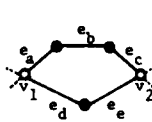
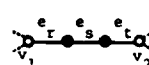
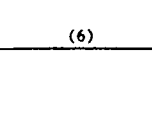
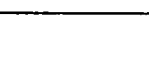
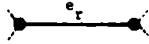
FIGURE 2.6

to include the failure probabilities of their end vertices. To extend the notation to unreliable vertices, the reliability of any vertex $v_i \notin K$ is denoted by p_i where the subscript i is a numeral. Where p_i must be redefined in Table 2.2, p_i' is used to signify this redefined value. Now the modified series reduction is obvious. Let two edges in G_K , $e_a = (v_1, v_2)$ and $e_b = (v_2, v_3)$ be candidates for a series reduction, i.e., $\deg(v_2) = 2$ and $v_2 \notin K$. Then e_a and e_b may be replaced by a single edge $e_c = (v_1, v_3)$ to obtain G'_K where $p_c = p_a p_b p_2$ so that $R(G_K) = R(G'_K)$. The reliabilities of v_1 and v_3 , if defined, are unchanged.

The modified polygon-to-chain reductions are not so obvious but may be derived by the method of Theorem 2.1. It is only necessary to condition first on whether or not unreliable vertices are working and include the additional variables p_1' and p_2' corresponding to the unreliable vertices v_1 and v_2 . Conditioning on a vertex, say v_1 , failing simply induces the graph $G_K - v_1$ while conditioning on the vertex working just gives us back G_K but with v_1 having perfect reliability as in the normal case. Note that in two of the reductions there is only one non-K-vertex and in these cases there is only one unreliable vertex and only one additional variable since K-vertices are always assumed completely reliable. Table 2.2 lists the extended polygon-to-chain reductions. Their proofs are omitted since they are simple extensions of Theorem 2.1.

TABLE 2.2
Polygon-to-Chain Reductions with Unreliable Vertices

Note: Darkened vertices represent K-vertices

Polygon Type	Chain Type	Reduction Formulas	New Edge Reliabilities
 <p>(1)</p>		$\alpha = p_1 p_2 q_a p_b q_c$ $\beta = p_1 p_2 p_a q_b q_c$ $\delta = p_1 p_2 p_a p_b p_c \left(1 + \frac{q_a}{p_a} + \frac{q_b}{p_b} + \frac{q_c}{p_c}\right)$	$\phi = q_1 p_1 p_a$ $\theta = p_1 q_2 p_b$ $\Omega = \frac{(\alpha + \delta + \phi)(\beta + \delta + \theta)}{\delta}$ $p_r = \frac{\delta}{\alpha + \delta}$ $p_s = \frac{\delta}{\beta + \delta}$ $p_1' = \frac{\alpha + \delta}{\alpha + \delta + \phi}$ $p_2' = \frac{\beta + \delta}{\beta + \delta + \theta}$
 <p>(2)</p>		$\alpha = p_2 q_a p_b q_c$ $\beta = p_2 p_a q_b q_c$ $\delta = p_2 p_a p_b p_c \left(1 + \frac{q_a}{p_a} + \frac{q_b}{p_b} + \frac{q_c}{p_c}\right)$	$\phi = q_2 p_a$ $\theta = p_2 q_1 p_b$ $\Omega = \frac{(\alpha + \delta)(\beta + \delta + \phi)}{\delta}$ $p_r = \frac{\delta}{\alpha + \delta}$ $p_s = \frac{\delta}{\beta + \delta}$ $p_2' = \frac{\beta + \delta}{\alpha + \delta + \phi}$
 <p>(3)</p>		$\alpha = p_2 (p_a q_b q_c p_d + q_a p_b p_c q_d + q_a p_b q_c p_d)$ $\beta = p_a q_b p_c q_d$ $\delta = p_a p_b p_c p_d \left(1 + \frac{q_a}{p_a} + \frac{q_b}{p_b} + \frac{q_c}{p_c} + \frac{q_d}{p_d}\right)$	$\phi = q_2 p_3 p_c$ $\theta = p_2 q_1 p_a p_d$ $\Omega = \frac{(\alpha + \delta)(\beta + \delta + \phi)}{\delta}$
 <p>(4)</p>		$\alpha = p_1 p_2 q_a p_b q_c p_d$ $\beta = p_1 p_2 (p_a q_b q_c p_d + q_a p_b p_c q_d)$ $\delta = p_1 p_2 p_a p_b p_c p_d \left(1 + \frac{q_a}{p_a} + \frac{q_b}{p_b} + \frac{q_c}{p_c} + \frac{q_d}{p_d}\right)$	$\phi = q_1 p_1 p_b p_d$ $\theta = p_1 q_1 p_a p_c$ $\Omega = \frac{(\alpha + \delta)(\beta + \delta + \phi)}{\delta}$ $p_r = \frac{\gamma}{\alpha + \gamma}$ $p_s = \frac{\gamma}{\beta + \gamma}$ $p_t = \frac{\gamma}{\delta + \gamma}$
 <p>(5)</p>	 <p>See Note</p>	$\alpha = p_1 p_2 q_a p_b p_c q_d$ $\beta = p_1 p_2 p_a p_b p_c q_d$ $\delta = p_1 p_2 p_a p_b p_c p_d \left(1 + \frac{q_a}{p_a} + \frac{q_b}{p_b} + \frac{q_c}{p_c} + \frac{q_d}{p_d}\right)$	$\phi = q_1 p_2 p_b p_c$ $\theta = p_1 q_2 p_a p_b$ $\Omega = \frac{(\alpha + \gamma + \phi)(\beta + \gamma + \theta)}{\gamma^2}$ $p_1' = \frac{\alpha + \gamma}{\alpha + \gamma + \phi}$ $p_2' = \frac{\beta + \gamma}{\beta + \gamma + \theta}$
 <p>(6)</p>		$\alpha = p_1 p_2 q_a p_b p_c q_d p_e$ $\beta = p_1 p_2 (p_a q_b p_c (p_d q_e + q_d p_e) + p_b (q_a p_c p_d q_e + p_a q_c q_d p_e))$ $\delta = p_1 p_2 p_a p_b p_c p_d p_e \left(1 + \frac{q_a}{p_a} + \frac{q_b}{p_b} + \frac{q_c}{p_c} + \frac{q_d}{p_d} + \frac{q_e}{p_e}\right)$	$\phi = q_1 p_2 p_b p_c p_e$ $\theta = p_1 q_2 p_a p_b p_d$ $\Omega = \frac{(\alpha + \gamma + \phi)(\beta + \gamma + \theta)}{\gamma^2}$
 <p>(7)</p>		$\alpha = p_1 p_2 q_a p_b p_c q_d p_e p_f$ $\beta = p_1 p_2 (p_a q_b p_c (q_d p_e p_f + p_d q_e p_f + p_d p_e q_f) + p_a p_b p_c p_d (p_d q_e + q_d p_e) + q_a p_b p_c p_d (q_e p_f + p_e q_f))$ $\delta = p_1 p_2 p_a p_b p_c p_d p_e p_f \left(1 + \frac{q_a}{p_a} + \frac{q_b}{p_b} + \frac{q_c}{p_c} + \frac{q_d}{p_d} + \frac{q_e}{p_e} + \frac{q_f}{p_f}\right)$	$\phi = q_1 p_2 p_b p_c p_e p_f$ $\theta = p_1 q_2 p_a p_b p_d p_e$ $\Omega = \frac{(\alpha + \gamma + \phi)(\beta + \gamma + \theta)}{\gamma^2}$ $p_r = (p_b + p_1 p_2 p_a q_b p_c p_d) / \Omega$ $\Omega = p_b + p_1 p_2 p_a q_b p_c$ <p>Note: For $K = 2$ chain is</p>  <p>Compare Theorem 3.3f</p>

Chapter 3

Triconnected Decomposition for the K-Terminal Problem

3.1 Introduction

Suppose we have a graph G_K with separating pair $\{u, v\}$. Recall from the previous chapter that this means that G_K can be partitioned into two graphs \dot{G}_K and \ddot{G}_K such that $G_K = \dot{G}_K \cup \ddot{G}_K$, $|\dot{E}| \geq 2$, $|\ddot{E}| \geq 2$, $\dot{E} \cap \ddot{E} = \emptyset$ and $\dot{V} \cap \ddot{V} = \{u, v\}$. (The K-vertices in the above expressions are not part of the definition of a separating pair but are necessary for explaining the results of this chapter. We define $\dot{K} = K \cap \dot{V}$ and $\ddot{K} = K \cap \ddot{V}$.) It has been known for some time that for certain configurations of the K-vertices, it is possible to replace \dot{G}_K or \ddot{G}_K with a single edge $e = (u, v)$ so as to yield a reliability-preserving reduction on G_K . (Birnbaum and Esary [1965], Rosenthal [1974]) For other configurations however, this is not possible. Rosenthal [1974] was forced to employ "hyper-edges" to handle these other cases where a hyper-edge could have three states: working, failed and system-failed. Unfortunately, such a reduction destroys the simplicity of the graphical model.

In this chapter, we present a method by which a subgraph between a separating pair may always be replaced by a chain of one, two or three edges. Although the proofs are quite general, we may normally assume that the subgraph to be replaced is a "pseudo-triconnected component," that is, the subgraph would be a triconnected component if all its chains were replaced by single edges. This is a natural way to look at the problem since a graph can be efficiently decomposed, with respect to its separating pairs, into its triconnected components using the algorithm of Hopcroft and Tarjan [1973]. These components are of minimal size and result in

computation being minimized. Hagstrom [1980] shows how to use the decomposition tree produced by the Hopcroft and Tarjan algorithm to compute reliability in the two-terminal case, and it is a simple extension to use those methods in the K-terminal case. Computational complexity of a problem becomes proportional to the complexity of the largest triconnected component in the graph if this decomposition is used in a one-shot manner on a given graph. Even greater advantage should be obtained if this decomposition were used as one of the reductions employed in a general factoring algorithm.

In the following sections we first discuss triconnected decomposition and how it should be applied to the K-terminal reliability problem. Next, we prove that it is always possible to replace a subgraph associated with a separating pair $\{u, v\}$, with a chain $\chi(u, v)$ consisting of one, two or three edges. These proofs do not indicate how to compute the necessary chain-edge reliabilities and multiplication factors for the reductions, so, in the following section, we show that the necessary calculations can be carried out by computing the K-terminal reliability of one, two, three or four graphs defined from the triconnected component alone. In the final section, the results are extended to the case where vertices may be unreliable.

3.2 Triconnected Decomposition of a Graph

A necessary concept for this discussion is that of the **vertex connectivity** of a graph $G=(V,E)$. The definition varies somewhat from author to author and from application to application; we offer the definition from Tutte [1966] which is used in his discussion of triconnected decomposition. Let V_0 be a subset of the vertices of V and let $G-V_0$ be the graph obtained from G by deleting all vertices $v \in V_0$ and all edges incident to those vertices. Suppose V_0 is the smallest set of vertices such that $G-V_0$ is disconnected or $|V-V_0|=1$. Then G is said to be **k-connected** if $|V_0| \geq k$. We are using the terms "biconnected" and "triconnected" to mean 2-connected and 3-connected, respectively.

The basic irreducible element of a graph in the K-terminal reliability problem is, in some sense, not the edge but the chain. As we will show in the next section, any subgraph between a separating pair can be replaced by a chain in a reliability-preserving reduction. However, if

this subgraph is already a chain, our reduction is not terribly useful. With this in mind, let G^+ be G with all chains replaced by single edges. Then G is said to be **pseudo-triconnected** if G^+ is triconnected. If G is pseudo-triconnected, then we may be able to decompose G along its separating pairs as described below, but it will be of no use as far as calculating reliability goes. Even if the graph is not pseudo-triconnected we may separate out chains needlessly. However, this does not increase the asymptotic complexity of the decomposition algorithm so we ignore the problem.

Suppose we have a graph G which is biconnected but not triconnected. We hope that G is not pseudo-triconnected; that would be boring. G can be uniquely decomposed into a tree of **triconnected components** consisting of simple cycles, bonds (three or more edges in parallel) and triconnected graphs with no parallel edges. These components contain artificial "virtual" edges as will be described below, following Hopcroft and Tarjan [1973]. Let $\{u, v\}$ be a separating pair of G , define \dot{G} and \ddot{G} as in the first section and let $\dot{e}=(u, v)$ and $\ddot{e}=(u, v)$ be two virtual edges which will be considered to be brothers. G can be decomposed into two split graphs $\dot{G}+\dot{e}$ and $\ddot{G}+\ddot{e}$. Each split graph can be split again and again until all split graphs consist of triple bonds (bonds with three edges), triangles and triconnected graphs. These graphs are called **split components** of G and are not necessarily unique. Now define a **merging operation** to be the reverse of a splitting operation, including the deletion of both virtual-edge brothers. By merging triple bonds with each other and by merging triangles with each other, a unique set of bonds, simple cycles and triconnected graphs will be created. These triconnected components will be arranged in tree-fashion, with nodes representing components and branches representing splitting operations. Here the terms "node" and "branch" are used to refer to the vertices and edges of a graph which is not a probabilistic graph. If desired, the original graph G can be reconstructed by merging any leaf component with the component containing the brother of its virtual edge and recursively repeating this process on the reduced tree until no more virtual edges remain.

Lichtenstein [1981] uses a different definition of "triconnected" and indicates how to con-

struct the decomposition tree directly, without resorting to merging. His definition of triconnected is: "A graph G is triconnected if there exist at least three node disjoint paths between every pair of vertices." This means that bonds are also considered triconnected and the decomposition uniquely decomposes G into triconnected graphs and **exactly biconnected** graphs. An exactly biconnected graph has exactly two node disjoint paths between every pair of vertices and is thus a simple cycle. Using either outlooks on the problem, the decomposition can be carried out very efficiently, in time which is linear in the number of vertices and edges of the graph.

Now, the leaf nodes of the decomposition tree will be of the form $\dot{G} + \dot{e}$ where $\dot{e} = (u, v)$ is a virtual edge and \dot{G} is a subgraph of G such that $\dot{G} \cup \dot{G} = G$, $\dot{E} \cap \dot{E} = \emptyset$ and $\dot{V} \cap \dot{V} = \{u, v\}$. So (u, v) is a separating pair and, as will be proven below, \dot{G}_K may be replaced by chain in a reliability-preserving reduction. This can be carried out in the tree by deleting \dot{G} and \dot{e} and replacing the brother of \dot{e} with the derived chain. After this replacement either the decomposition tree is reduced to a single node or there remain leaves which can be replaced in a reliability-preserving fashion as just described. Of course, some degree-2 or series reductions may also be admitted and these may be performed as they appear. After repeated reductions, the tree is eventually reduced to a single node which is a graph with no virtual edges. This graph is either a simple cycle or a pseudo-triconnected graph, either of which may admit some series or degree-2 reductions. After making any final reductions, the reliability of the original graph, $R(G_K)$, is given by the reliability of the reduced graph times the product of all the multiplication factors derived in any of the reductions. If the remaining graph has been reduced to two edges in parallel then we are essentially done. If not, then we must apply some general method to compute the reliability of the remaining graph.

3.3 Triconnected-Component-to-Chain Reductions

In this section we simply show that any subgraph with separating pair $\{u, v\}$ can be replaced, in a reliability-preserving reduction, by a chain consisting of one, two or three edges between u and v . This is a general result which need not be applied to triconnected components derived from the decomposition described in the previous section. However,

decomposing the graph as completely as possible and making these reductions would normally be the most computationally efficient procedure. The proof of the triconnected-component-to-chain reduction is based largely on state enumeration and does not indicate, except in the first case, how to calculate the chain-edge reliabilities and multiplication factor efficiently. That discussion appears at the end of this section.

Theorem 3.1: Let G_K be a nonseparable graph with separating pair $\{u, v\}$ such that $G_K = \dot{G}_K \cup \ddot{G}_K$ where $\dot{V} \cap \ddot{V} = \{u, v\}$, $|\dot{E}| \geq 2$, $|\ddot{E}| \geq 2$ and $\dot{E} \cap \ddot{E} = \emptyset$. Then \dot{G}_K may be replaced by a chain $\chi(u, v)$ consisting of one, two or three edges to obtain a reliability-preserving reduction of G_K to $G'_K = \ddot{G}_K \cup \chi(u, v)$.

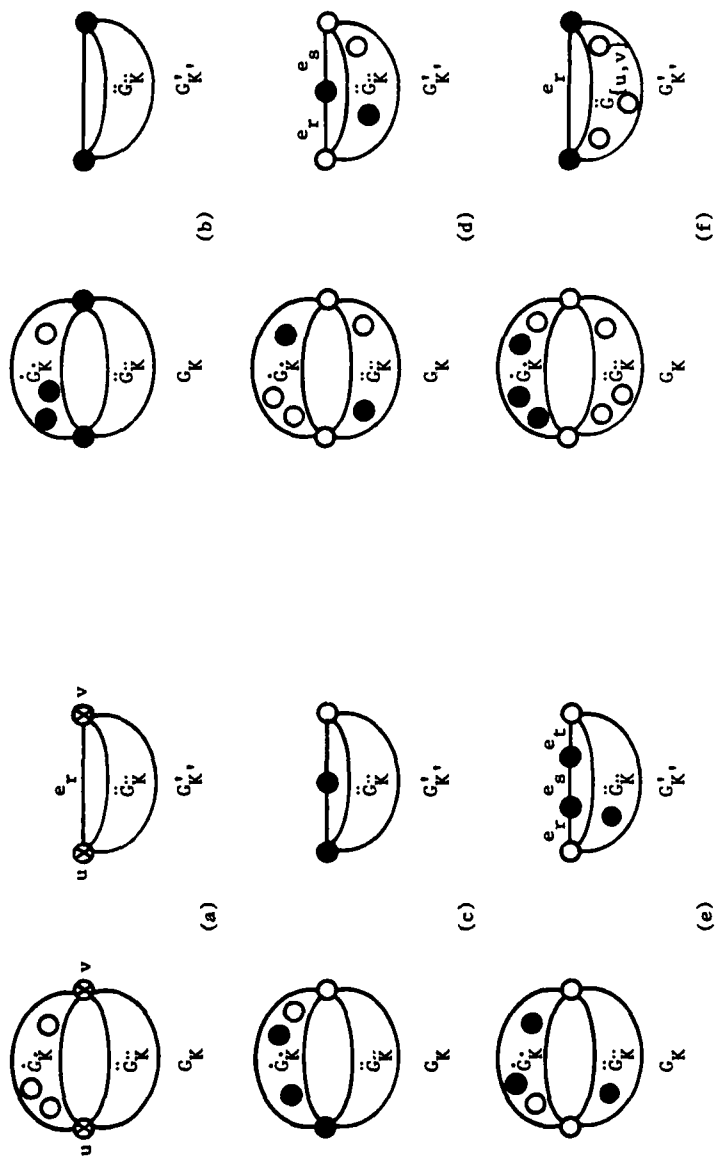
The proof below is divided into six parts depending on the configuration of the K-vertices in G . The first two parts are cases which can be found in the literature; the last four parts are new. Parts (d), (e) and (f) are similar to (c) and will be somewhat abbreviated. Figures 3.1-3.5 illustrate the proof.

Proof of Theorem 3.1:

(a) Suppose $\dot{K} - u - v = \emptyset$. (Vertices u and v may or may not be elements of K and \dot{K} as indicated by crosses within the vertices, Figure 3.1a.) Birnbaum and Esary [1965] establish, and it is fairly obvious, that \dot{G} may be replaced by a single edge $e_r = (u, v)$ with $p_r = R(\dot{G}_{(u,v)})$, so that $R(G_K) = R(G'_K)$. See Figure 3.1a.

(b) Suppose $u, v \in K$ and $\dot{K} - u - v \neq \emptyset$. Rosenthal [1974] proves that \dot{G} may be replaced by a single edge $e_r = (u, v)$ with $p_r = R(\dot{G}_K)/\Omega$ so that $R(G_K) = \Omega R(G'_K)$. Ω is defined in terms of \dot{G}_K alone, by $\Omega = \text{Prob}(\text{All } w \in \dot{K} \text{ can communicate with } u \text{ or } v)$. In other words, Ω is the probability that the states of the edges in \dot{G} do not imply that G_K has, *a priori*, failed. See Figure 3.1b.

(c) Suppose $u \in K$, or $v \in K$ but not both, and $\dot{K} - u - v \neq \emptyset$. Assume, without loss of generality, that $u \in K$ and $v \notin K$. As in the proof of Theorem 2.1, we can condition on the compound states of the edges in \dot{G} so that



Tricconnected-Component-to-Chain Reductions

FIGURE 3.1

$$R(G_K) = \sum_{F \in \mathcal{F}} \text{Prob}(F) R(G_K|F) \quad (3.1)$$

where \mathcal{F} is the set of all $2^{|E|}$ states of \dot{G} , F indicates one of those states, and $\text{Prob}(F)$ indicates the probability of the state F occurring, i.e., $\text{Prob}(F) = \prod_{e_i \in E} p_i^{z_i} q_i^{1-z_i}$ where $z_i=1$ if e_i works in F and $z_i=0$ if e_i fails in F . Now \mathcal{F} can be divided into four mutually exclusive and exhaustive classes F_i , $i=0,1,2,3$:

$$F_0 = \{F \in \mathcal{F}: \text{In } \dot{G}_K|F, \text{ some } w \in \dot{K} \text{ can communicate with neither } u \text{ nor } v.\}$$

$$F_1 = \{F \in \mathcal{F}: \text{In } \dot{G}_K|F, \text{ all } w \in \dot{K} \text{ can communicate with } u \text{ or } v \text{ but not both and there is at least one such } w \text{ which can communicate with } v.\}$$

$$F_2 = \{F \in \mathcal{F}: \text{In } \dot{G}_K|F, \text{ all } w \in \dot{K} \text{ can communicate with } u \text{ but not with } v.\}$$

$$F_3 = \{F \in \mathcal{F}: \text{In } \dot{G}_K|F, \text{ all } w \in \dot{K} \text{ can communicate with both } u \text{ and } v.\}$$

Since G_K is nonseparable, F_1 , F_2 and F_3 are nonempty. $R(G_K|F)=0$ for all $F \in F_0$ and may therefore be disregarded in the remainder of the proof.

By the definition of the F_i , $i=1,2,3$, any state $F \in F_i$ will induce the graph $G_{i,K}$, as shown in Figure 3.2. Letting

$$\alpha = \sum_{F \in F_1} \text{Prob}(F) \quad \beta = \sum_{F \in F_2} \text{Prob}(F) \quad \delta = \sum_{F \in F_3} \text{Prob}(F) \quad (3.2)$$

and by grouping and eliminating terms in Equation 3.1 we have

$$R(G_K) = \alpha R(G_{1,K}) + \beta R(G_{2,K}) + \delta R(G_{3,K}) \quad (3.3)$$

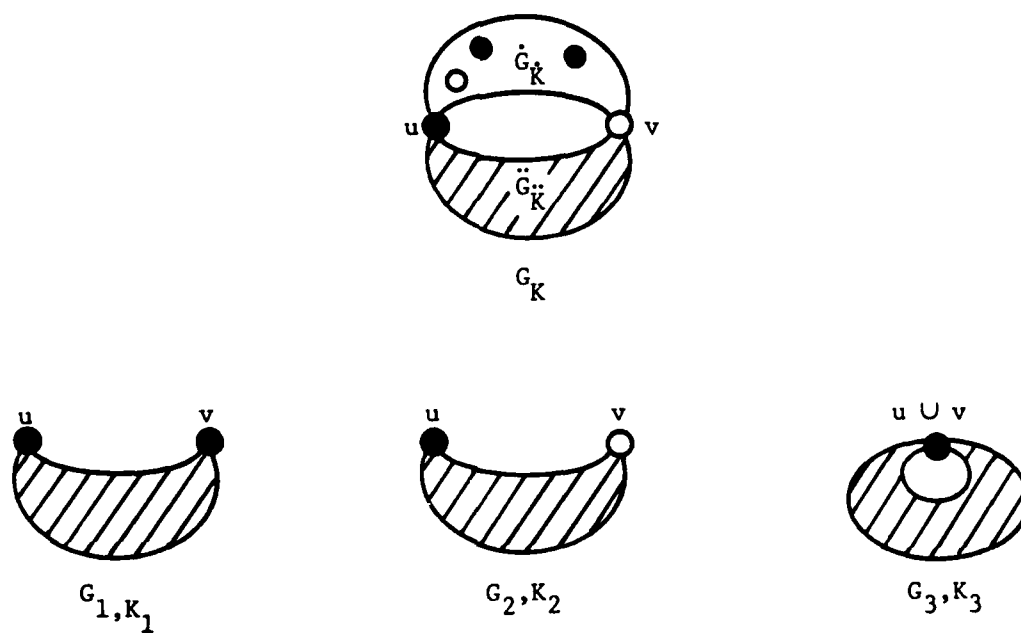
Now let G'_K be $\dot{G}_K \cup \chi(u, v)$ where the chain $\chi(u, v)$ contains edges e_r and e_s as shown on the right hand side of Figure 3.1c. With these edges we associate p_r , p_s and Ω defined as follows:

$$p_r = \frac{\delta}{\alpha + \delta} \quad p_s = \frac{\delta}{\beta + \delta} \quad \Omega = \frac{(\alpha + \delta)(\beta + \delta)}{\delta} \quad (3.4)$$

Conditioning on the edges e_r and e_s in G'_K induces the same non-failed subgraphs (Figure 3.2)

$G_{1,K}$, $G_{2,K}$, $G_{3,K}$, with respective probabilities, $q_r p_s$, $p_r q_s$, and $p_r p_s$. Therefore

$$\begin{aligned} \Omega R(G'_K) &= \Omega q_r p_s R(G_{1,K}) + \Omega p_r q_s R(G_{2,K}) + \Omega p_r p_s R(G_{3,K}) \\ &= \alpha R(G_{1,K}) + \beta R(G_{2,K}) + \delta R(G_{3,K}) \\ &= R(G_K) \end{aligned}$$



Non-failed Graphs Induced by Factoring on \dot{G}_K
for Theorem 3.1c

FIGURE 3.2

Thus, any component of the specified type may be replaced by a chain as shown in Figure 3.1c with p_r , p_s and Ω defined via Equations 3.2 and 3.4, and this will be a reliability-preserving reduction.

The above reduction is valid if $\dot{K}=K$ but we might not want to approach the problem in that manner. $\dot{K}=K$ implies that $\ddot{K}-u=\emptyset$, and consequently it is possible to replace \dot{G} with a single edge $e_r=(u,v)$ with reliability $p_r=R(\dot{G}_{(u,v)})$ to yield the reliability-preserving reduction described in (a) above. If we do wish to operate on \dot{G}_K first however, the reduction can be simplified. If $\dot{K}=K$ then $R(G_{2,K_2})=R(G_{3,K_3})=1$ and thus,

$$R(G_K) = \alpha R(G_{1,K_1}) + \beta + \delta \quad (3.5)$$

We needn't replace \dot{G} with a chain at all but rather find α , β and δ and then $R(G_{1,K_1})$. From the definitions of α and β we see that $R(\dot{G}_K)=\beta+\delta$. Computation of α is discussed in Theorem 3.2. Equation 3.5 may be thought of as an extended reliability-preserving reduction of the form $R(G_K)=\Omega_1+\Omega_2 R(G'_K)$ where the graph replacing the subgraph \dot{G} is null.

(d) Suppose $u, v \notin K$ and $\dot{K}=\{w\}$. Furthermore, assume that $\ddot{K} \neq \emptyset$ since otherwise $R(G_K) \equiv 1$. The possible states of \dot{G}_K may be partitioned into four classes F_i , $i=0,1,2,3$:

$$F_0 = \{\underline{F} \in F: \text{In } \dot{G}_K | \underline{F}, w \text{ can communicate with neither } u \text{ nor } v.\}$$

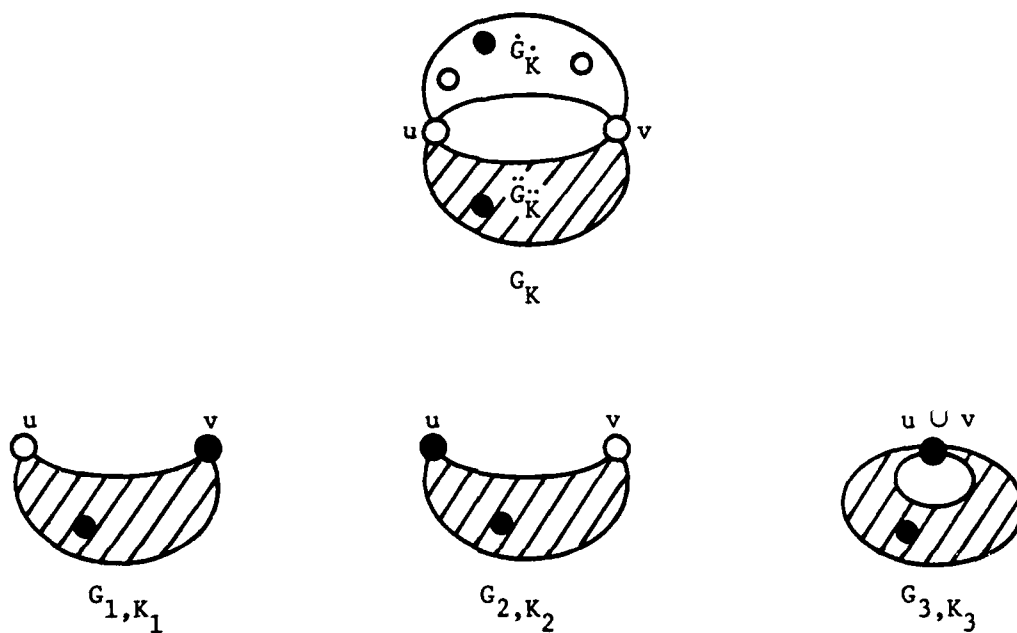
$$F_1 = \{\underline{F} \in F: \text{In } \dot{G}_K | \underline{F}, w \text{ can communicate with } v \text{ but not with } u.\}$$

$$F_2 = \{\underline{F} \in F: \text{In } \dot{G}_K | \underline{F}, w \text{ can communicate with } u \text{ but not with } v.\}$$

$$F_3 = \{\underline{F} \in F: \text{In } \dot{G}_K | \underline{F}, w \text{ can communicate with both } u \text{ and } v.\}$$

Now define α , β and δ as in Equations 3.2, but with respect to the above F_i . Noting again that $R(G_K | \underline{F})=0$ for all $\underline{F} \in F_0$, the remainder of this proof is identical to that in (c) except that the induced subgraphs are different. See Figure 3.3. Defining p_r , p_s , and Ω as in Equations 3.4 makes the transformation in Figure 3.1d a reliability-preserving reduction.

(e) Suppose $u, v \notin K$, $|K| \geq 2$ and $\ddot{K} \neq \emptyset$ (Figure 3.1e). The set of possible states F for \dot{G}_K may be partitioned into five classes F_i , $i=0,1,2,3,4$:



Non-failed Graphs Induced by Factoring on \dot{G}_K
for Theorem 3.1d

FIGURE 3.3

$F_0 = \{\underline{F} \in \mathcal{F}: \text{In } \dot{G}_K | \underline{F}, \text{ some } w \in \dot{K} \text{ can communicate with neither } u \text{ nor } v.\}$

$F_1 = \{\underline{F} \in \mathcal{F}: \text{In } \dot{G}_K | \underline{F}, \text{ all } w \in \dot{K} \text{ can communicate with } v \text{ but not with } u.\}$

$F_2 = \{\underline{F} \in \mathcal{F}: \text{In } \dot{G}_K | \underline{F}, \text{ all } w \in \dot{K} \text{ can communicate with either } u \text{ or } v \text{ but not both, and at least one vertex } x \in \dot{K} \text{ communicates with } u, \text{ and at least one vertex } y \in \dot{K} \text{ communicates with } v.\}$

$F_3 = \{\underline{F} \in \mathcal{F}: \text{In } \dot{G}_K | \underline{F}, \text{ all } w \in \dot{K} \text{ can communicate with } u \text{ but not with } v.\}$

$F_4 = \{\underline{F} \in \mathcal{F}: \text{In } \dot{G}_K | \underline{F}, \text{ all } w \in \dot{K} \text{ can communicate with both } u \text{ and } v.\}$

Again, $R(G_K | \underline{F}) = 0$ for all $\underline{F} \in F_0$. Any state $\underline{F} \in F_i$, $i=1,2,3,4$ will induce the subgraph $G_{i,K}$ from G_K as shown in Figure 3.4. Letting

$$\alpha = \sum_{\underline{F} \in F_1} \text{Prob}(\underline{F}) \quad \beta = \sum_{\underline{F} \in F_2} \text{Prob}(\underline{F}) \quad \delta = \sum_{\underline{F} \in F_3} \text{Prob}(\underline{F}) \quad \gamma = \sum_{\underline{F} \in F_4} \text{Prob}(\underline{F}) \quad (3.6)$$

we have

$$R(G_K) = \alpha R(G_{1,K}) + \beta R(G_{2,K}) + \delta R(G_{3,K}) + \gamma R(G_{4,K})$$

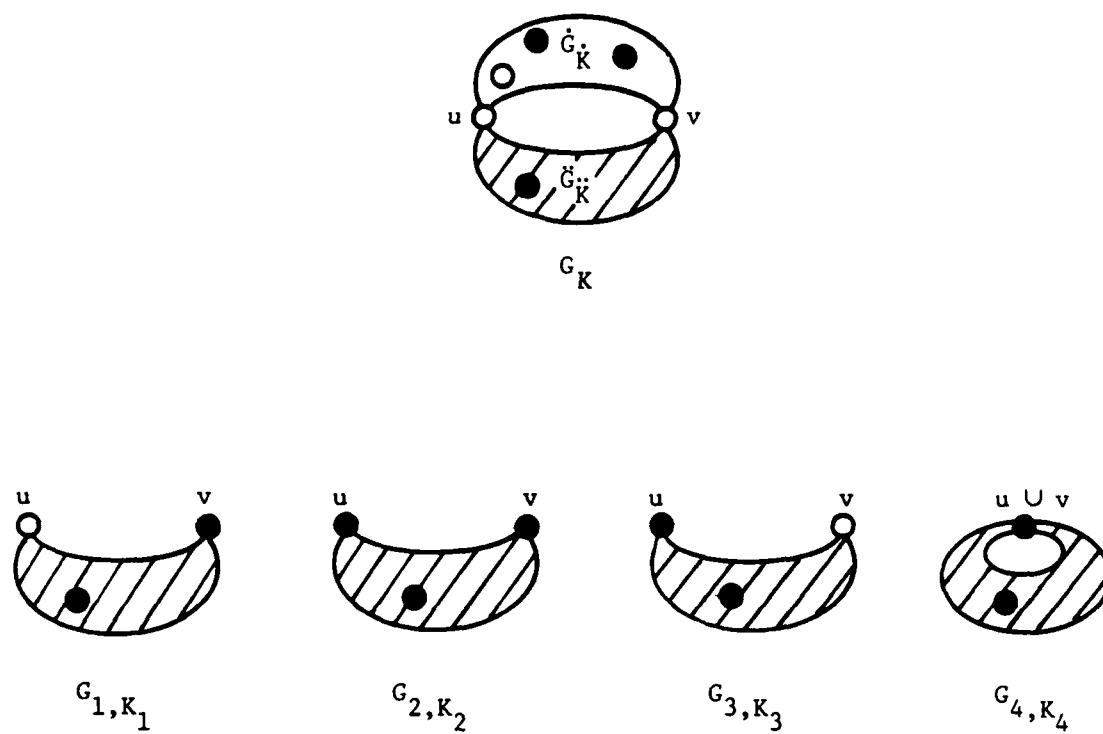
Now let G'_K be G_K with \dot{G}_K replaced by the chain $\chi(u,v)$ comprising edges e_r , e_s and e_t as shown in Figure 3.1e. Define p_r , p_s , p_t and Ω :

$$\begin{aligned} p_r &= \frac{\gamma}{\alpha + \gamma} & p_s &= \frac{\gamma}{\beta + \gamma} \\ p_t &= \frac{\gamma}{\delta + \gamma} & \Omega &= \frac{(\alpha + \gamma)(\beta + \gamma)(\delta + \gamma)}{\gamma^2} \end{aligned} \quad (3.7)$$

Conditioning on the edges e_r , e_s and e_t in G'_K gives, in a similar fashion to (c):

$$\begin{aligned} \Omega R(G'_K) &= \Omega p_r q_s p_t R(G_{1,K}) + \Omega q_r p_s p_t R(G_{2,K}) + \\ &\quad \Omega p_r p_s q_t R(G_{3,K}) + \Omega p_r p_s p_t R(G_{4,K}) \\ &= \alpha R(G_{1,K}) + \beta R(G_{2,K}) + \delta R(G_{3,K}) + \gamma R(G_{4,K}) \\ &= R(G_K) \end{aligned}$$

Thus, any subgraph of the specified type may be replaced by chain as shown in Figure 3.1e with p_r , p_s , p_t and Ω obtained from Equations 3.6 and 3.7, and this will be a reliability-preserving reduction.



Non-failed Graphs Induced by Factoring on \dot{G}_K for Theorem 3.1e

FIGURE 3.4

(f) Suppose $u, v \in K$, $|K| \geq 2$ and $\ddot{K} = \emptyset$ (Figure 3.1f). The set of states F of \dot{G}_K may be partitioned into three classes $F_i, i=0,1,2$:

$F_0 = \{\underline{F} \in F: \text{At least one } w \in \dot{K} \text{ can communicate with neither } u \text{ nor } v \text{ and fails to communicate with at least one other vertex } y \in \dot{K}.\}$

$F_1 = \{\underline{F} \in F: \text{All } w \in \dot{K} \text{ can communicate with } u \text{ or } v \text{ but not both, and at least one such vertex communicates with } u \text{ and another with } v.\}$

$F_2 = \{\underline{F} \in F: \text{All } w \in \dot{K} \text{ can communicate.}\}$

As before, $R(G_K | \underline{F}) = 0$ for all $\underline{F} \in F_0$. Any state in F_1 will induce the graph G_{1,K_1} (Figure 3.5). States of F_2 will induce graphs $G_{2,K_2}, G_{3,K_3}, G_{4,K_4}$ and G_{5,K_5} . Letting

$$\alpha = \sum_{\underline{F} \in F_1} \text{Prob}(\underline{F}) \quad \beta = \sum_{\underline{F} \in F_2} \text{Prob}(\underline{F})$$

and noting that $R(G_{2,K_2}) = R(G_{3,K_3}) = R(G_{4,K_4}) = R(G_{5,K_5}) = 1$ since $\ddot{K} = \emptyset$, we have

$$R(G_K) = \alpha R(G_{1,K_1}) + \beta R(G_{2,K_2})$$

Now let G'_K be G_K with \dot{G}_K replaced by a single edge $e_r = (u, v)$ and where $K' = \{u, v\}$. Defining

$$p_r = \frac{\beta}{\alpha + \beta} \quad \Omega = \alpha + \beta$$

we have

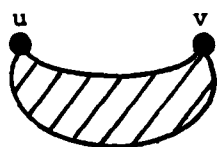
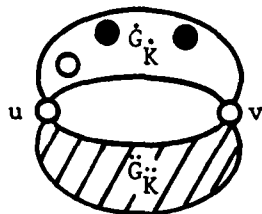
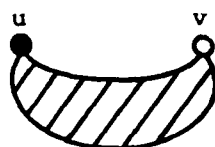
$$\begin{aligned} \Omega R(G'_K) &= \Omega q_r R(G_{1,K_1}) + \Omega p_r R(G_{2,K_2}) \\ &= \alpha R(G_{1,K_1}) + \beta R(G_{2,K_2}) \\ &= R(G_K) \end{aligned}$$

Thus, we see that \dot{G}_K can be replaced by a single edge. Of course, $R(G_{2,K_2}) = 1$ and therefore

$$R(G_K) = \alpha R(G_{1,K_1}) + \beta = \alpha R(\ddot{G}_{\{u,v\}}) + \beta$$

This, as mentioned in (c) above, is an extended reliability-preserving reduction where we replace a subgraph with a null graph.

Any triconnected component with associated K -vertices fits into one of the above six categories and thus, any such component may be replaced by a chain of one, two or three edges in a reliability-preserving reduction. \square

 G_{1,K_1}  G_{2,K_2}  G_{3,K_3}  G_{4,K_4}  G_{5,K_5}

Non-failed Graphs Induced by Factoring on \dot{G}_K for Theorem 3.1f

FIGURE 3.5

It now remains to be shown how the reductions of Theorem 3.1 can be carried out in an efficient manner, or at least in a manner which is more efficient than state enumeration. We will, in Theorem 3.2, prove that only one, two, three or four K -terminal reliability problems defined on \dot{G} need be solved in order to compute chain-edge reliabilities and multiplication factor Ω in any of the reductions.

In the following discussion, the notation $G[uv]$ is used to indicate the graph G in which vertices u and v have been coalesced to form the vertex $u' = u \cup v$. Lemma 3.1 below is used to simplify the proof of Theorem 3.2. The proof of the lemma is stated in terms of a general graph G_K but, in fact, will be applied in Theorem 3.2 to the graph \dot{G}_K , the subgraph of G_K .

Lemma 3.1: Let G_K be a graph with two specified vertices u and v such that $K - u - v \neq \emptyset$. Then

$$\text{Prob}(\text{All } w \in K - u - v \text{ can communicate with } u \text{ or } v) = R(G_K'[uv])$$

where $K' = K - u - v + u'$.

Proof: Since G and $G[uv]$ have the same edge sets, any state \underline{F} of $G[uv]$ is a state of G and vice versa. What we must show is that every state \underline{F} in which $G_K'[uv]|\underline{F}$ is working implies, in a one-to-one correspondence, that all $w \in K$ can communicate with u or v in $G_K|\underline{F}$.

Let \underline{F} be state of G_K in which all $w \in K$ can communicate with u or v , i.e., there exists a path of working edges from every vertex $w \in K$ to u or v in $G_K|\underline{F}$. Every such path must correspond to a path of $G_K'[uv]|\underline{F}$ from $w \in K'$ to u' so that $G_K'[uv]|\underline{F}$ is working.

Conversely, if \underline{F} is a working state of $G_K'[uv]$, then all $w \in K'$ can communicate with u' , i.e., there exists a path of working edges in $G_K'[uv]|\underline{F}$ from each $w \in K'$ to u' . But any such path corresponds to a path in $G_K|\underline{F}$ from w to u or v . Thus any working state of $G_K'[uv]$ corresponds to a state in G_K in which every vertex $w \in K$ can communicate with u or v . \square .

Now, without further ado, we state and prove Theorem 3.2.

Theorem 3.2: Let G_K be a graph with subgraph \dot{G}_K and associated separating pair $\{u, v\}$ as in Theorem 3.1. Then the subgraph-to-chain reduction can be carried out, i.e., all chain-edge reli-

abilities and the factor Ω computed, by solving one, two, three or four K-terminal reliability problems defined on \dot{G} .

Proof: Cases (a)-(f) here correspond directly to cases (a)-(f) in Theorem 3.1.

(a) ($\dot{K}-u-v=\emptyset$) $R(G_K)$ is affected by \dot{G} only in whether or not u and v can communicate through \dot{G} . Therefore, \dot{G} is replaced by the edge $e_r=(u,v)$ with $p_r=R(\dot{G}_{\{u,v\}})$. Only a single two-terminal reliability problem defined on \dot{G} must be solved here.

(b) ($u,v \in K$ and $\dot{K}-u-v \neq \emptyset$) Here, two problems must be solved to derive p_r and Ω . By Lemma 3.1,

$$\begin{aligned}\Omega &= \text{Prob}(\text{All } w \in \dot{K}-u-v \text{ can communicate with } u \text{ or } v) \\ &= R(\dot{G}_K'[uv])\end{aligned}$$

where $K'=\dot{K}-u-v+u'$. So Ω can be computed by coalescing vertices u and v in \dot{G}_K , making the coalesced vertex u' a new K-vertex and computing the reliability of the resulting graph. Since $p_r=R(\dot{G}_K)/\Omega$, p_r can be obtained after computing Ω and the reliability of \dot{G}_K .

(c) ($u \in K$, $v \notin K$ and $K-u \neq \emptyset$) All that must be shown here is how to compute α , β , and δ , since p_r , p_s , and Ω can be derived from those values. From the definition of α , β and δ we have

$$R(\dot{G}_{K+v}) = \delta \tag{3.8}$$

$$R(\dot{G}_K) = \beta + \delta \tag{3.9}$$

and by Lemma 3.1

$$R(\dot{G}_{K-u+u'}[uv]) = \alpha + \beta + \delta \tag{3.10}$$

Solving these equations for α , β and δ is trivial and thus, this reduction can be carried out by solving three reliability problems defined on \dot{G} .

If $|\dot{K}-u|=1$ in the above case, the next equation is also valid and may be used to replace Equation 3.10:

$$R(\dot{G}_{K-u+v}) = \alpha + \delta$$

This equation has the advantage that no coalescing of vertices has to be performed and the complexity of computing reliability on the original component may be less. For example, \dot{G}

may have a series-parallel structure which is destroyed by coalescing u and v .

(d) ($u, v \notin K$ and $\dot{K} = \{w\}$) Analogous to Equations 3.8, 3.9 and 3.10, but with the definitions of α , β , and δ from Theorem 3.1(d), we have

$$R(\dot{G}_{K+u+v}) = \delta$$

$$R(\dot{G}_{K+u}) = \beta + \delta$$

$$R(\dot{G}_{K+v}) = \alpha + \delta$$

which are easily solved. Note that in lieu of any one of the above equations,

$$R(\dot{G}_{K+u}\{uv\}) = \alpha + \beta + \delta$$

could also be used.

(e) ($u, v \notin K$, $|\dot{K}| \geq 2$ and $\ddot{K} \neq \emptyset$) From the definitions of α , β , δ and γ in Theorem 3.1(e), and using Lemma 3.1, the following relations hold:

$$R(\dot{G}_{K+u+v}) = \gamma$$

$$R(\dot{G}_{K+u}) = \delta + \gamma$$

$$R(\dot{G}_{K+v}) = \alpha + \gamma$$

$$R(\dot{G}_{K+u}\{uv\}) = \alpha + \beta + \delta + \gamma$$

These too are easily solved.

(f) ($u, v \notin K$, $|\dot{K}| \geq 2$ and $\ddot{K} = \emptyset$) From Theorem 3.1(f) and Lemma 3.1 we have

$$R(\dot{G}_K) = \beta$$

$$R(\dot{G}_{K+u}\{uv\}) = \alpha + \beta$$

and therefore this reduction can be carried out by solving two reliability problems defined on \dot{G}_K .

Therefore, by solving one, two, three or four reliability problems on \dot{G} , \dot{G}_K may be replaced by a chain of one, two or three edges so as to yield a reliability-preserving reduction. \square

3.4 Extension to Unreliable Vertices

The results of the previous two sections may be generalized to handle separating-pair vertices u and v which may be unreliable. Other vertices in the graph may be unreliable also without affecting any of the results. We will not provide proofs here but just give the necessary

information to carry out the reductions. As in section 2.6, the notation will be changed to emphasize the fact that the separating-pair vertices are unreliable. We let v_1 correspond to u and v_2 correspond to v , and indicate respective vertex reliabilities by p_1 and p_2 for the original graph G_K and p_1' and p_2' for the reduced graph G'_K . The reduction process is naturally more complex here but in all cases (a)-(f), the new chain to be introduced is the same as in Theorem 3.1 (a)-(f). In order to compute all parameters for these reductions it may be necessary to evaluate the reliability of up to six different graphs defined from \dot{G} , including the reliability of that graph with v_1 or v_2 deleted.

Theorem 3.3: Let G_K be a graph whose non-K-vertices may be unreliable. Also, let G_K be a nonseparable graph with separating pair $\{v_1, v_2\}$ such that $G_K = \dot{G}_K \cup \ddot{G}_K$, $\dot{V} \cap \ddot{V} = \{v_1, v_2\}$, $\dot{E} \cap \ddot{E} = \emptyset$, $|\dot{E}| \geq 2$, and $|\ddot{E}| \geq 2$. Then \dot{G}_K may be replaced by a chain $\chi(v_1, v_2)$ consisting of one, two or three edges to obtain a reliability-preserving reduction of G_K to $G'_K = \ddot{G}_K \cup \chi(v_1, v_2)$. Furthermore, all chain-edge reliabilities, vertex reliabilities and any multiplication factor can be obtained by computing the reliability of no more than six problems defined on \dot{G} .

The reductions below depend on computing the reliability of a graph in which one or two vertices have been conditioned to be working or failed. When a vertex v_i is failed in graph \dot{G}_K , it and its incident edges are deleted and we use the notation $\dot{G}_K - v_i$, which was introduced earlier. We use the notation $\dot{G}_K | v_i$ when vertex v_i is working, i.e., is perfectly reliable. The vertex $v' = v_1 \cup v_2$ is always assumed to be perfectly reliable. Cases (a)-(f) below correspond directly to cases (a)-(f) in Theorems 3.1 and 3.2.

(a) ($\dot{K} - v_1 - v_2 = \emptyset$) Here, $p_r = R(\dot{G}_{|u,v})$ and if $v_1 \notin K$ then $p_1' = p_1$ and if $v_2 \notin K$ then $p_2' = p_2$. (Recall that any K-vertex is considered completely reliable so if $v_1 \in K$ or $v_2 \in K$ here or below, we do not explicitly refer to the reliabilities of these vertices.)

(b) ($v_1, v_2 \in K$ and $\dot{K} - v_1 - v_2 \neq \emptyset$) There is no change in this reduction since $u, v \in K$.

(c) ($v_1 \in K$, $v_2 \notin K$ and $\dot{K} - u - v \neq \emptyset$) New multiplication factor and reliabilities are given by

$$\begin{aligned} p_r &= \frac{\delta}{\alpha + \delta} & p_s &= \frac{\delta}{\beta + \delta} \\ p_2' &= \frac{\beta + \delta}{\beta + \delta + \phi} & \Omega &= \frac{(\alpha + \delta)(\beta + \delta + \phi)}{\delta} \end{aligned}$$

Parameters α , β , δ and ϕ can be obtained from the following equations:

$$\begin{aligned} R(\dot{G}_{K+v_2}|v_2)p_2 &= \delta \\ R(\dot{G}_K|v_2)p_2 &= \beta + \delta \\ R(\dot{G}_{K-v_1+v_2}[v_1v_2])p_2 &= \alpha + \beta + \delta \\ R(\dot{G}_K - v_2)q_2 &= \phi \end{aligned}$$

(d) ($u, v \notin K$ and $\dot{K} = \{w\}$) The new multiplication factor and reliabilities are given by

$$\begin{aligned} p_r &= \frac{\delta}{\alpha + \delta} & p_s &= \frac{\delta}{\beta + \delta} \\ p_1' &= \frac{\alpha + \delta}{\alpha + \delta + \phi} & p_2' &= \frac{\beta + \delta}{\beta + \delta + \phi} \\ \Omega &= \frac{(\alpha + \delta + \phi)(\beta + \delta + \phi)}{\delta} \end{aligned}$$

Parameters α , β , δ and ϕ can be obtained from the following equations:

$$\begin{aligned} R(\dot{G}_{K+v_1+v_2}|v_1, v_2)p_1p_2 &= \delta \\ R(\dot{G}_{K+v_1}|v_1, v_2)p_1p_2 &= \beta + \delta \\ R(\dot{G}_{K+v_2}|v_1, v_2)p_1p_2 &= \alpha + \delta \\ R(\dot{G}_{K+v_2}-v_1|v_2)q_1p_2 &= \phi \\ R(\dot{G}_{K+v_1}-v_2|v_1)p_1q_2 &= \theta \end{aligned}$$

(e) ($u, v \notin K$, $|\dot{K}| \geq 2$ and $\dot{K} \neq \emptyset$) The new multiplication factor and reliabilities are given by

$$\begin{aligned} p_r &= \frac{\gamma}{\alpha + \gamma} & p_s &= \frac{\gamma}{\beta + \gamma} \\ p_1 &= \frac{\gamma}{\delta + \gamma} & p_1' &= \frac{\alpha + \gamma}{\alpha + \gamma + \phi} \\ p_2' &= \frac{\delta + \gamma}{\delta + \gamma + \theta} & \Omega &= \frac{(\alpha + \gamma + \phi)(\beta + \gamma)(\delta + \gamma + \theta)}{\gamma^2} \end{aligned}$$

Parameters α , β , δ and γ can be obtained from the following equations:

$$\begin{aligned} R(\dot{G}_{K+v_1+v_2}|v_1, v_2)p_1p_2 &= \gamma \\ R(\dot{G}_{K+v_1}|v_1, v_2)p_1p_2 &= \delta + \gamma \\ R(\dot{G}_{K+v_2}|v_1, v_2)p_1p_2 &= \alpha + \gamma \\ R(\dot{G}_{K+v_2}[v_1v_2])p_1p_2 &= \alpha + \beta + \delta + \gamma \\ R(\dot{G}_{K+v_2}-v_1|v_2)p_1q_2 &= \phi \\ R(\dot{G}_{K+v_1}-v_2|v_1)p_1q_2 &= \theta \end{aligned}$$

(f) ($v_1, v_2 \notin K$, $|K| \geq 2$ and $K \neq \emptyset$) In this reduction v_1 and v_2 become K-vertices and thus we do not explicitly define p_1' and p_2' . The multiplication factor and edge reliability for this reduction are given by

$$p_r = \frac{\beta + \phi}{\alpha + \beta + \phi} \quad \Omega = \alpha + \beta + \phi$$

Parameters α , β and ϕ may be obtained from the following equations:

$$R(\dot{G}_{K+v_1+v_2} | v_1, v_2) p_1 p_2 = \beta$$

$$R(\dot{G}_{K+v} [v_1 v_2]) p_1 p_2 = \alpha + \beta$$

$$R(\dot{G}_{K-v_1} | v_2) q_1 p_2 + R(\dot{G}_{K-v_2} | v_1) p_1 q_2 + R(\dot{G}_{K-v_1-v_2}) q_1 q_2 = \phi \quad \square$$

Chapter 4

Polygon-to-Chain Reductions and Factoring

4.1 Introduction

In this chapter we investigate the complexity of computing K -terminal reliability when using a factoring algorithm coupled with simple reductions and the new polygon-to-chain reductions of Chapter 2. We place one restriction on edge selection and establish an optimal edge-selection strategy for a factoring algorithm when $2 \leq |K| \leq 5$ or $|V| - 2 \leq |K| \leq |V|$. The amount of work performed by this algorithm is shown to be proportional to a combinatorial invariant of G called the minimum domination. When the restriction on edge selection is removed, algorithmic complexity is essentially unchanged for $|V| - 2 \leq |K| \leq |V|$ but, for $|K|$ in the lower range, minimum domination turns out to be only a tight upper bound on complexity. These new results are significant from a computational viewpoint since minimum domination may be exponentially smaller than the computational bound attainable without using polygon-to-chain reductions.

A factoring algorithm for computing K -terminal reliability is essentially an implicit state enumerator. We recursively apply Equation 1.3, making reductions and attempt to avoid zero reliability graphs so as to enumerate explicitly as few states as possible. We stop when the reliability of the reduced graphs can be computed directly without further factoring. A recursive algorithm of this type is a backtrack procedure because a partial solution is extended until completely solved; the algorithm backtracks to the last created partial solution and extends that solution; and continues on in this way until all partial solutions are solved, and thus the whole problem is solved. Chang [1981] and Johnson [1982] discuss, in detail, backtrack algorithms

tailored to network reliability.

The backtrack procedure can be represented by a binary search structure--binary because at each factoring step of the algorithm, exactly two subproblems are created. If the complexity of the reductions and edge-selection strategies employed is polynomially bounded, then the complexity of the whole algorithm is essentially proportional to the number of leaf nodes in the backtrack structure since this number tends to be exponential in the size of the problem. Satyanarayana and Chang [1981] analyzed the complexity of computing $R(G_K)$ using a factoring algorithm with series and parallel reductions. They found that the number of leaves in the associated backtrack structure is at least equal to the "domination" of G_K , a combinatorial invariant denoted $D(G_K)$. Using a simple edge-selection strategy, they further showed that it is possible to create a backtrack structure which has exactly $D(G_K)$ leaves. Therefore, a factoring algorithm using this strategy and series and parallel reductions is optimal.

Chang [1981] goes one step further and analyzes a factoring algorithm for the all-terminal problem which utilizes parallel and degree-2 reductions. He proves that the optimal algorithm generates a backtrack structure with the number of leaves equal to another combinatorial invariant called the "minimum domination" of G . The minimum domination, denoted $M(G)$, is the minimum of $D(G_K)$ over all $K \subseteq V$, with $|K|=2$. (Johnson [1982] points out that $M(G)$ is equivalent to a combinatorial invariant on the graphic matroid of G called the "Crapo beta-invariant.") Chang and Johnson also provide simple edge-selection strategies which insure that the algorithm is optimal. $D(G_V)$ may be exponentially larger than $M(G)$ so this minimum domination result is important. Unfortunately, the analysis does not extend to the general K -terminal case.

One would hope that adding polygon-to-chain reductions to the arsenal of reductions would significantly reduce the computational complexity of a factoring algorithm. We already know from the exposition in Chapter 2 that this will be true in some cases. Since the polygon-to-chain reductions can be bought for little more than the cost of simple reductions alone, only two facts must be established in order to develop a good factoring algorithm: (i) The edge-

selection strategy devised can be implemented efficiently; and (ii) this strategy will produce fewer leaves in the backtrack structure than other possible strategies. We will do this by generalizing the minimum domination results of Chang [1981] pertaining to the all-terminal problem. As will be seen, we are only successful for $2 \leq |K| \leq 5$ and $|V| - 2 \leq |K| \leq |V|$.

4.2 Preliminaries

In this section, we define a few pertinent concepts and detail useful properties of the combinatorial invariant, minimum domination. A K -tree of a graph G_K is any minimal graph which connects all the K -vertices of G_K . It is a tree which covers all K -vertices and whose degree-1 vertices are all in K . An edge is irrelevant if it does not lie in some K -tree of G_K .

A formation of G_K is a set of K -trees whose union is G_K . Letting N_o be the number of odd cardinality formations of G_K and letting N_e be the number of even cardinality formations of G_K , then the domination of G_K is defined by

$$D(G_K) = |N_o - N_e|$$

Clearly, $D(G_K)$ is a combinatorial invariant of G_K .

The minimum domination of a graph G is defined by

$$M(G) = \min_{K: |K| \geq 2} D(G_K)$$

The following properties of $M(G)$ are established or implicit in Chang [1981].

Property 4.1

- (a) $M(G) = M(G - e) + M(G \cdot e)$.
- (b) $M(G)$ is invariant under series and parallel replacements.
- (c) $M(G) > 0$ if and only if G has no self-loops and is biconnected.
- (d) $M(G) = 1$ if and only if G is single edge or a biconnected series-parallel graph.

Property 4.1a is a factoring theorem for minimum domination, and along with the other properties, it will enable us to analyze the amount of work a network reliability factoring algorithm requires. In the next section we briefly specify a framework for a network reliability

factoring algorithm and show how Property 4.1 may be utilized to find the optimal algorithm for the all-terminal problem within the given framework. A similar tack is taken in the analysis of the K-terminal problem in Section 4.4.

4.3 Optimal Factoring Algorithms

The following algorithm which employs the recursive function REL, describes a general framework for computing network reliability via factoring.

Algorithm 4.1

MAIN

Input: A non-separable graph G_K with associated edge probabilities.

Output: K-terminal reliability of G_K .

- (1) $R \leftarrow \text{REL}(G_K)$
- (2) Print("R(G_K) is" R) and STOP.

End of MAIN.

REL(G_K)

Input: Graph G_K with associated edge reliabilities.

Output: Returns the value R which is the K-terminal reliability of G_K .

- (1) $M \leftarrow 1$.
- (2) If G_K is disconnected, Return(0).
- (3) If $|K|=1$, Return (1).
- (4) Until no more reliability-preserving reductions can be made
 - (a) Make any desired reduction on G_K to obtain G'_K and Ω such that $R(G_K) = \Omega R(G'_K)$.
 - (b) $M \leftarrow M\Omega$.
 - (c) $G_K \leftarrow G'_K$.
- (5) If G_K is a K-tree, Return($M \prod_{e_i \in E} p_i$).
- (6) Select an edge e_i .
- (7) $R \leftarrow M(p_i \text{REL}(G_K * e_i) + q_i \text{REL}(G_K - e_i))$
- (8) Return(R).

End of REL

The exact complexity of the above algorithm will of course depend on what sort of reductions are used and how much work is required to select an edge for factoring. However, the

reductions and edge-selection strategies are generally of polynomial complexity, and so we usually worry only about the number of calls to REL since this number tends to be exponential. Each call to REL corresponds to a node of the related binary search structure. The total number of these nodes is proportional to the number of leaf nodes in the structure so we say that the algorithm's complexity is proportional to the number of leaf nodes in the associated binary search structure.

Suppose we use Algorithm 4.1 with a particular set of reductions and a particular edge-selection strategy and show that the number of leaves in the associated binary search structure must be equal to some combinatorial invariant of the graph such as domination or minimum domination. If we can show that any other edge-selection strategy yields a number of leaves which is greater than the invariant, then the algorithm must be optimal. We show this, subject to one restriction, when $2 \leq |K| \leq 5$ or when $|V| - 2 \leq |K| \leq |V|$. By then removing the restriction, we find that the number of leaves in the backtrack structure is essentially unchanged for $|V| - 2 \leq |K| \leq |V|$, but may actually be less than the value of the invariant for $2 \leq |K| \leq 5$; thus, we have an algorithm which is demonstrably better than previously available algorithms and whose complexity is bounded but which is not necessarily optimal in all cases.

We note that there is no known way of computing $D(G_K)$ or $M(G)$ other than using a factoring algorithm or other exponentially complex procedure unless G possesses some special structure. Consequently, the complexity results do not normally yield any *a priori* estimate on the computational requirements of a specific problem. Below, we discuss how domination and minimum domination have been used to find optimal factoring algorithms.

Satyanarayana and Chang [1981] show how a network reliability factoring algorithm using series and parallel reductions must create at least $D(G_K)$ leaf nodes in its search structure. If the algorithm never factors so as to create graphs with irrelevant edges or graphs which are disconnected, then the number of leaves will be exactly equal to $D(G_K)$ and the algorithm will be optimal. An optimal edge-selection strategy for this algorithm is easily implemented. We will not go into detail about this particular algorithm but rather describe how, for the all-

terminal problem, an algorithm can be found which will optimally produce only $M(G)$ leaves. This discussion, a synthesis of Chang [1981] and Johnson [1982], is the basis for the complexity results of Section 4.4.

Consider using Algorithm 4.1 with parallel and degree-2 reductions in order to compute $R(G_V)$. Self-loops will never be created because they can only occur if parallel edges are not reduced. Suppose that we can always find an edge to factor on which never creates cutvertices in either of the induced graphs. If this is possible, then the minimum domination of all the induced graphs will always be greater than or equal to 1 by Properties 4.1c and 4.1d. Using this fact and Properties 4.1a and 4.1b, the domination of all leaf nodes must sum to $M(G)$. Since the domination of each leaf node graph must be 1, the total number of leaf nodes will be $M(G)$.

It is not hard to see that $M(G)$ is the minimal number of nodes possible in this algorithmic framework. Let G' be some graph intermediate in the factoring algorithm. G' admits no parallel or degree-2 reductions and $M(G') > 1$. Suppose we select an edge e for factoring such that either $G' * e$ or $G' - e$ is separable. Only one of these induced graphs can be separable since $M(G') = M(G' * e) + M(G' - e)$ and if both were separable, we would conclude that $M(G') = 0$ which would be a contradiction. Now assume without loss of generality that $G' * e$ is separable. Associated with $G' * e$ is at least one leaf node in the algorithm's search structure and associated with $G' - e$ are at least $M(G')$ leaf nodes. In this case, the algorithm must create at least $M(G') + 1$ leaf nodes below G' and therefore at least $M(G) + 1$ leaf nodes in total. Thus an algorithm which creates cutvertices cannot create the minimal number of nodes possible in its search structure. The algorithm which never creates separable graphs will be optimal if the edge selection strategy can be carried out efficiently. This is easily done.

Chang's edge-selection strategy amounts to factoring on any edge which is not adjacent to any vertex of a separating pair. If G is triconnected then any edge will do. Otherwise, an appropriate edge can be identified by examining the triconnected components and separating pairs of G and therefore, the edge-selection can be carried out in linear time (See Section 3.2).

Johnson also shows that if G is biconnected, there must exist a split component of G which has only one virtual edge and at least five real edges and that any edge of such a component may be selected for factoring.

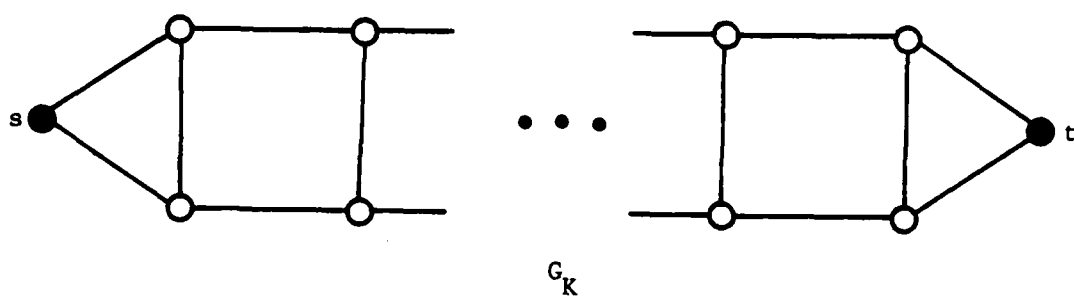
4.4 New Complexity Results for the K -terminal Problem

Now let's take a look at the complexity of a factoring algorithm utilizing simple reductions and polygon-to-chain reductions. In order to simplify the initial discussion, one restriction is first applied: Never factor on an edge e of G_K such that $|K'|=1$ in $G_K \cdot e$. It will be shown that for $2 \leq |K| \leq 5$ or $|V|-2 \leq |K| \leq |V|$, the optimal restricted algorithm will generate $M(G)$ leaves in its associated binary search structure and that a simple, linear-time edge-selection strategy will achieve this. These results, while not completely general, are significant from a computational point of view. Consider the graph of Figure 4.1. If we use the factoring algorithm of Satyanarayana and Chang [1981] which employs series and parallel reductions, the complexity of computing $R(G_K)$ is proportional to $D(G_K)=2^{(|E|-2)/3}$. With the addition of polygon-to-chain reductions, the complexity of the optimal algorithm is proportional to $M(G)=1$.

Results are basically unchanged for $|V|-2 \leq |K| \leq |V|$ when the restriction on edge selection is removed. However, when $2 \leq |K| \leq 5$ it may be possible to produce fewer than $M(G)$ leaves in backtrack search structure by factoring so as to give $|K|=1$ in an induced graph. $M(G)$ then becomes a tight upper bound on computational complexity. Actually, instead of factoring on such edges, we will define an extended reliability-preserving reduction which deletes any edge connecting two K -vertices when $|K|=2$ and show that this reduction cannot increase complexity.

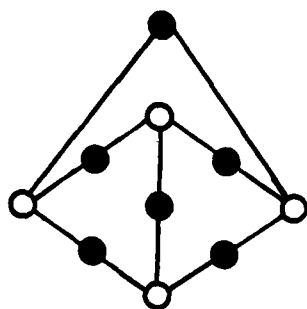
Preliminary results:

It is not hard to explain why the results of this section are limited to such unusual values of $|K|$. The optimal restricted algorithm depends on always being able to find an edge of G to factor on such that both $G \cdot e$ and $G - e$ are biconnected. Figures 4.2a and 4.2b show minimal pseudo-triconnected graphs with $|K|=6$ and $|K|=|V|-3$ respectively. Note that no matter



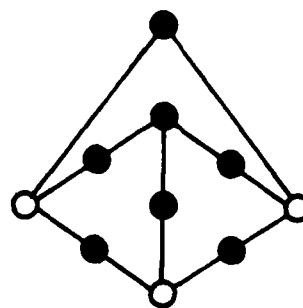
Graph with $D(G_k) = 2^{(|E|-2)/3}$ and $M(G) = 1$

FIGURE 4.1



$$|K| = 6$$

(a)



$$|K| = |V| - 3$$

(b)

Minimal Pseudo-Triconnected Graphs with $|K|$
Out of Given Range

FIGURE 4.2

which edge is selected for factoring, $G-e$ must have a cutvertex. The following lemma is necessary for proving that it is always possible to find a suitable edge for factoring if $|K|$ is within the given range.

Lemma 4.1: Suppose G is biconnected and has no series or parallel edges but is not triconnected. Then there exist two split components of G each containing at least five real edges and only one virtual edge.

Proof: Since G is biconnected but not triconnected, it must contain at least one separating pair. Let $\{u, v\}$ be the separating pair which allows us to partition G into two graphs $G^{(1)}$ and $G^{(2)}$ with $G = G^{(1)} \cup G^{(2)}$, $E^{(1)} \cap E^{(2)} = \emptyset$, $V^{(1)} \cap V^{(2)} = \{u, v\}$, $|E^{(1)}| \geq 2$, $|E^{(2)}| \geq 2$, and such that $E^{(1)}$ is minimal. Letting $e^{(1)} = (u, v)$ be a virtual edge, $G^{(1)} + e^{(1)}$ is a split component of G since it cannot be split any further. Since G had no series or parallel edges, $|V^{(1)}| \geq 4$ and $|E^{(1)}| \geq 5$ (Satyanarayana and Chang [1981]).

Next, consider $G^{(2)} + e^{(2)}$ where $e^{(2)} = (u, v)$ is a virtual edge. If $G^{(2)} + e^{(2)}$ is triconnected it is the other split component we were looking for and we are done. Otherwise it must be biconnected. Partition $G^{(2)} + e^{(2)}$ by a separating pair $\{x, y\}$ such that $G^{(2)} + e^{(2)} = G^{(3)} \cup G^{(4)}$, etc., but $e^{(2)} \notin E^{(3)}$ and $E^{(3)}$ is minimal. Since $e^{(2)} \notin E^{(3)}$, $G^{(3)}$ contains no parallel edges and by construction, it contains no series edges. As argued before, then $|V^{(3)}| \geq 4$ and $|E^{(3)}| \geq 5$. Letting $e^{(3)} = (x, y)$, be a virtual edge, we see that $G^{(3)} + e^{(3)}$ is a split component of G and the proof is complete. \square

Johnson [1982] proves this next helpful lemma.

Lemma 4.2: If G has no series or parallel edges and is biconnected but not triconnected, then $G \cdot e$ and $G - e$ will both be biconnected if e is any real edge of one of the split components of G as specified in Lemma 4.1.

Our new complexity results for the K-terminal reliability problem will follow directly from the following tedious theorem.

Theorem 4.1: Let G_K be a nontrivial, biconnected graph which admits no simple or polygon-

to-chain reductions. If $2 \leq |K| \leq 5$ or $|V| - 2 \leq |K| \leq |V|$ then there exists an edge $e \in E$ such that $M(G^*e) > 0$ and $M(G-e) > 0$.

Proof: G_K being nontrivial, biconnected and admitting no reductions, implies that $M(G) > 1$. Therefore, by Property 4.1c, we need only show that an edge $e \in E$ exists such that G^*e and $G-e$ are both biconnected. In the rest of the proof, we let G^+ be G with all chains of length two and three replaced by single edges, and we let $K^+ = \{v: v \in K, \deg(v) > 2\}$.

Case 1a: G^+ is triconnected and $2 \leq |K| \leq 5$. This implies that $|E^+| \geq 6$, $|V^+| \geq 4$ and there can be at most five chains containing degree-2 K -vertices. Thus, there exists an edge $e \in E^+$ which corresponds directly to an edge $e \in E$ as opposed to corresponding to a chain. Since G^+ is triconnected, G^+e and G^+-e must both be (at least) biconnected. These graphs may be obtained via series and parallel replacements from G^*e and $G-e$, respectively, and thus, by Properties 4.1b and 4.1c, we have $M(G^*e) = M(G^+e) > 0$ and $M(G-e) = M(G^+-e) > 0$.

Case 1b: G^+ is triconnected and $|V| - 2 \leq |K| \leq |V|$. We need to find an edge $e \in E^+$ which corresponds directly to an edge $e \in E$. It will suffice to find $e^+ = (u, v)$ where $u, v \in K^+$. Such an edge cannot correspond to a chain of length two or three in G , since otherwise G_K would admit a degree-2 reduction contrary to assumption.

Now, $|V^+| \geq 4$ since G^+ is non-trivial and triconnected. Select any two vertices $x, y \in K^+$. By Menger's well-known theorem, there exist three node disjoint paths from x to y since G is triconnected. At least one such path must contain only K -vertices since there are at most two vertices not in K^+ . Any edge $e = (u, v)$ in that path will have $u, v \in K^+$. Therefore $e \in E^+$ corresponds directly to an edge $e \in E$. The rest of the argument follows as in Case 1a and we have $M(G^*e) > 0$ and $M(G-e) > 0$.

Case 2a: G^+ is biconnected but not triconnected and $2 \leq |K| \leq 5$. By Lemma 4.1, there are at least two split components of G^+ with a total of at least ten edges. At least five of these edges must correspond directly to edges in G so select any such edge $e \in E^+$. By Lemma 4.2, G^+e and G^+-e are both biconnected and by Properties 4.1b and 4.1c we have $M(G^*e) = M(G^+e) > 0$ and $M(G-e) = M(G^+-e) > 0$.

Case 2b: G^+ is biconnected but not triconnected and $|V|-2 \leq |K| \leq |V|$. As in Case 1b, we need only show that there exists an edge $e=(u,v)$ in one of the split components of G^+ such that $u,v \in K^+$. Suppose there is no such edge. But there must exist an internal vertex $w \in K^+$ in one of the components. By "internal vertex" we just mean one of vertices of a component which is not a separating pair. Now, under the supposition, w can be adjacent to at most two vertices, neither of which can be K -vertices. This implies that $\deg(w) \leq 2$ which is a contradiction. Therefore there must exist an edge $e=(u,v) \in E^+$ such that $u,v \in K^+$. This edge is contained in one of the split components of G^+ and corresponds directly to an edge $e \in E$. Thus $G^+ \cdot e$ and $G^+ - e$ are biconnected and $M(G^+ \cdot e) = M(G^+ - e) > 0$ and $M(G - e) = M(G^+ - e) > 0$. \square

Theorem 4.1 essentially provides the edge-selection strategy for our restricted factoring algorithm: Always factor on an edge e such that $G \cdot e$ and $G - e$ are biconnected. A suitably modified and extended version of Hopcroft and Tarjan's triconnected decomposition algorithm could always find a proper edge in linear time. Therefore, in proving the optimality of the restricted factoring algorithm, we shall only be concerned with the number of leaves in the algorithm's backtrack search structure.

It would be handy to be able to say that minimum domination is invariant under simple reductions and polygon-to-chain reductions. However, minimum domination is defined with respect to G , not G_K , and consequently, such a statement would be somewhat inconsistent. Instead, we offer the following property for reference; its validity is a direct result of Property 4.1(b).

Property 4.2: Let G'_K be obtained from G_K by any number of simple or polygon-to-chain reductions. Then $M(G') = M(G)$.

Complexity of the restricted algorithm:

Given the restriction that no induced graph may ever have $|K|=1$, the next theorem proves the optimality of the edge-selection strategy described above.

Theorem 4.2: Let G_K be a biconnected graph with $2 \leq |K| \leq 5$ or $|V|-2 \leq |K| \leq |V|$. Then we can compute $R(G_K)$ in time proportional to $M(G)$ using a factoring algorithm coupled with

simple reductions and polygon-to-chain reductions and using an edge-selection strategy which never creates separable graphs. Furthermore, when restricting edge selection to those edges which never create induced graphs with $|K|=1$, this edge-selection strategy is optimal.

Proof: If $2 \leq |K| \leq 5$ initially, then $|K|$ remains within this range in the graphs created as the algorithm proceeds because neither factoring nor polygon-to-chain reductions ever create additional K -vertices, and because we do not allow factoring on edges so as to create graphs with $|K|=1$. (From Theorem 4.1 we know there are always at least four acceptable edges to factor on when $|K|=2$.) Similarly, no reductions or factoring can ever create more non- K -vertices so if $|V|-2 \leq |K| \leq |V|$ in the original graph, then this will also hold in any of the graphs encountered as the algorithm factors and reduces.

Since $|K|$ always remains in the given range, and since after factoring we do all possible simple and polygon-to-chain reductions, by Theorem 4.1, the algorithm will always be able to find an edge e such that $G \cdot e$ and $G - e$ are biconnected. Thus $M(G \cdot e) > 0$ and $M(G - e) > 0$ at each factoring step of the algorithm. Using this fact and Properties 4.1a and 4.2, the sum of the minimum dominations of all the leaf nodes of the algorithm's backtrack search structure must be $M(G)$. Since the minimum domination of each leaf node is 1, the total number of leaves must be $M(G)$. Because the reductions and edge selection at each step can be carried out in polynomial time, $R(G_K)$ can be computed in time proportional to $M(G)$.

The argument for optimality is the same as described for the all-terminal problem in Section 4.4. \square

Complexity of the unrestricted algorithm:

We discuss next how the complexity of the factoring algorithm changes when we allow factoring on an edge such that $|K|=1$ in one of the induced graphs. In an optimization problem, the value of the objective function can only remain the same or improve if a constraint is removed. Analogously, the unrestricted algorithm will be no more complex than the restricted algorithm. This situation deserves a little more attention than a simple statement, however. We analyze the change in complexity for $|V|-2 \leq |K| \leq |V|$ and $2 \leq |K| \leq 5$ separately below.

Theorem 4.3: The complexity of computing $R(G_K)$ is the same using the restricted and unrestricted algorithms when $|V|-2 \leq |K| \leq |V|$.

Proof: The algorithms can differ only when $|K'|=2$ in some induced graph G'_K . Let G'_K be such an induced graph. If $|K|=|V|$ or $|K|=|V|-1$ in the original graph G_K , then $|V'|=2$ or $|V'|=3$ in G'_K , $M(G')=1$ and the unrestricted factoring never comes into play since G'_K represents a leaf in the backtrack search structure for either algorithm.

If $|K|=|V|-2$ however, it is possible that $|K'|=2$, $|V'|=4$ and $M(G')>1$. However, this only occurs if G' is the complete graph on four vertices in which case $M(G')=2$. No matter which edge is chosen for factoring, the number of leaves below G' will be two. The restricted algorithm and unrestricted algorithm might not yield the same leaf graphs in their search structures but they must yield exactly the same number of leaves, namely $M(G)$. \square

There is an alternative method of looking at the unrestricted algorithm which simplifies the rest of the discussion. Consider a graph G_K such that $e_i=(u,v) \in E$ and $K=\{u,v\}$. Factoring on e_i gives

$$R(G_K) = p_i + q_i R(G_K - e_i)$$

But this is just an extended reliability-preserving reduction of the form $R(G_K) = \Omega_1 + \Omega_2 R(G'_K)$. It will be easier to discuss the complexity of the unrestricted factoring algorithm if this reduction is used rather than explicitly factoring on e_i . This reduction will be called the **trivial reduction** and would normally be applied whenever possible. Naturally, it can be carried out in linear time.

For the case where $|K|=|V|-2$, the trivial reduction can have the effect of reducing an induced graph G' with $M(G')=2$ to $G'-e_i$ with $M(G'-e_i)=1$. Thus the number of leaves in the unrestricted algorithm's search structure could be decreased by up to half over the number created by the restricted algorithm. However, we can say that the restricted algorithm is still optimal within a constant factor. Such is not the case when $2 \leq |K| \leq 5$ as will be demonstrated.

Consider the graph G_K of Figure 4.1. If we add the edge $e=(s,t)$, then the complexity of the restricted algorithm becomes proportional to $M(G+e)=2^{(|E|-3)/2}$. The complexity of the

unrestricted algorithm with the trivial reduction is proportional to $M(G)=1$. The two values vary by a decidedly non-constant factor.

It seems intuitively clear that the application of the trivial reduction cannot increase computational complexity. However, the situation is complicated a bit by the fact that deletion of the edge between the two K -vertices may create a separable graph. The factoring algorithm should then include biconnected decomposition so that the individual blocks can be identified and handled separately. The complexity of computing $R(G_K - e_i)$ will be proportional to the sum of the complexities of computing the reliability of the non-trivial blocks of $G_K - e_i$. Trivial blocks, i.e., bridges, effectively add no complexity since they can be identified and evaluated within the linear-time, biconnected decomposition. Note that the search structure may no longer be binary after a trivial reduction; rather, it will have as many branches below $G_K - e_i$ as there are non-trivial, biconnected components of $G_K - e_i$.

The final theorem of this dissertation says that application of the trivial reduction cannot increase the complexity of the factoring algorithm when $2 \leq |K| \leq 5$. The notation $L(G_K)$ is used to indicate the number of leaves in the unrestricted algorithm's search structure below a graph G_K .

Theorem 4.4: Let G_K be any biconnected graph with $2 \leq |K| \leq 5$. Then $L(G_K - e) \leq M(G)$.

Proof: The argument here is inductive on the number of edges in G . The statement is obviously true for graphs with six or fewer edges. Assume it is true for graphs with m or fewer edges and suppose $|E| = m+1$.

If $3 \leq |K| \leq 5$, the proof is trivial. Select an edge e for factoring which satisfies the unrestricted algorithm's edge-selection strategy. This edge is, of course, the same for both algorithms. Select any edge e such that G^*e and $G - e$ are both biconnected. This gives

$$\begin{aligned} L(G_K) &= L(G_K^*e) + L(G_K - e) \\ &\leq M(G^*e) + M(G - e) \text{ by the induction hypothesis} \\ &= M(G) \end{aligned} \tag{4.1}$$

If $|K|=2$ but the two K -vertices are not adjacent, the same argument holds.

Next suppose $|K|=2$, say $K=\{u, v\}$, and u and v are adjacent. Let $e=(u, v)$. The next step in the unrestricted algorithm is to reduce G_K by deleting edge e . If $G-e$ is biconnected, then

$$\begin{aligned} L(G_K) &= L(G_K - e) \\ &\leq M(G-e) \text{ by the induction hypothesis} \\ &= M(G) - M(G^*e) \\ &\leq M(G) \end{aligned} \tag{4.2}$$

The only difficult case arises when $G-e$ is separable.

If $G-e$ is separable, then G must be of the form $G^{(1)} \cup G^{(2)} \cup e$ as shown in Figure 4.3a. We add the zero-reliability edges $e^{(1)}=(u, w)$ and $e^{(2)}=(v, w)$ as shown in Figure 4.3b and first assume that $G^{(1)}+e^{(1)}$ and $G^{(2)}+e^{(2)}$ are both nontrivial, i.e., their minimum dominations are both greater than one. Then,

$$M(G) = M(G+e^{(1)}+e^{(2)}) \tag{4.3}$$

$$= M(G^{(1)}+e^{(1)})M(G^{(2)}+e^{(2)}) \tag{4.4}$$

$$\geq M(G^{(1)}+e^{(1)}) + M(G^{(2)}+e^{(2)}) \text{ by assumption of nontriviality}$$

$$\geq L((G^{(1)}+e^{(1)})_{\{u, w\}}) + L((G^{(2)}+e^{(2)})_{\{v, w\}})$$

$$= L(G^{(1)}_{\{u, w\}}) + L(G^{(2)}_{\{v, w\}})$$

$$= L(G_K - e)$$

$$= L(G_K)$$

Equation 4.3 may be easily derived by adding edges $e^{(1)}$ and $e^{(2)}$ to G , factoring on those edges and applying Property 4.1a to the resulting graphs. Equation 4.4 follows from the facts that (1) $D(G_K) = D(\dot{G}_K)D(\ddot{G}_K)$ if $G = \dot{G} \cup \ddot{G}$, $\dot{E} \cap \ddot{E} = \emptyset$, $\dot{V} \cap \ddot{V} = \{u, v\}$, $\dot{E} \geq 2$, $\ddot{E} \geq 2$ and $K = \{u, v\}$; and (2) $M(G) = D(G_{\{x, y\}})$ for any adjacent pair of vertices x and y . See Chang [1981].

One of $G^{(1)}$ or $G^{(2)}$ but not both may be a single edge. Without loss of generality, assume that $G^{(1)}$ is an edge e_1 . $R(G_K - e) = p_1 R(G^{(2)}_{\{v, w\}})$ so $L(G_K)$ is just $L(G^{(2)}_{\{v, w\}})$.

Therefore

$$M(G) = M(G^{(2)}+e^{(2)})$$

$$\geq L((G^{(2)}+e^{(2)})_{\{v, w\}}) \text{ by the induction hypothesis}$$

$$= L(G^{(2)}_{\{v, w\}})$$

$$= L(G_K) \quad \square$$

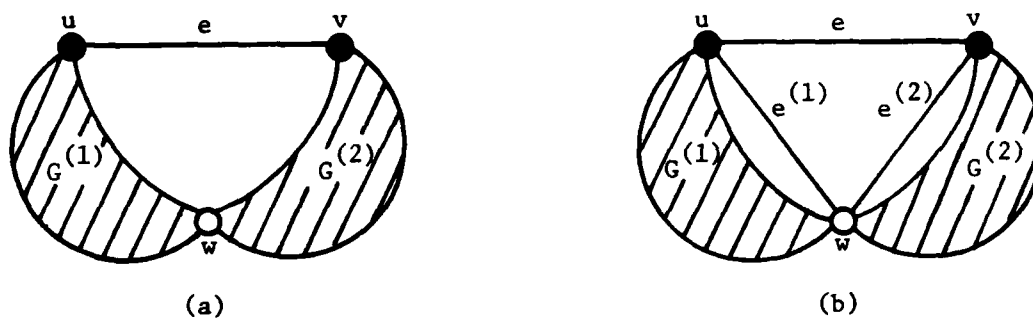


Illustration for Theorem 4.4

FIGURE 4.3

References

Michael O. Ball

- [1980] "Complexity of Network Reliability Computations," *Networks*, 10, 153-165.

M.O. Ball and J.S. Provan

- [1981] "The Complexity of Counting Cuts and of Computing the Probability that a Graph is Connected," Working Paper 81-002, University of Maryland at College Park.

R.E. Barlow and F. Proschan

- [1975] *Statistical Theory of Reliability*, Holt, Rinehart and Winston, New York, N.Y..

J.A. Buzzacott

- [1980] "A Recursive Algorithm for Finding Reliability Measures Related to the Connection of Nodes in a Graph," *Networks*, 10, 311-327.

Mark K. Chang

- [1981] "A Graph Theoretic Appraisal of the Complexity of Network Reliability Algorithms," Dept. of IEOR, University of California, Berkeley.

N. Deo

- [1974] *Graph Theory with Applications to Engineering and Computer Science*, Prentice-Hall, Englewood Cliffs, N.J..

R.J. Duffin

- [1965] "Topology of Series-Parallel Networks," *J. Math. Analysis and Applications*, 10, 303-318.

L. Fratta and U. Montanari

- [1973] "A Boolean Algebra Method for Computing the Terminal Reliability in a Communication Network," *IEEE Trans. Circuit Theory*, CT-20, 203-211.

M.R. Garey and D.S. Johnson

- [1979] *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Co., San Francisco, California.

Jane N. Hagstrom

- [1980] "Combinatoric Tools for Computing Network Reliability," Ph.D. Thesis, Dept. of IEOR, University of California, Berkeley.

E. Hänsler, G.K. McAuliffe and R.S. Wilkov

- [1974] "Exact Calculation of Computer Network Reliability," *Networks*, 4, 95-112.

Frank Harary

- [1969] *Graph Theory*, Addison-Wesley, Reading, Mass.

J.E. Hopcroft and R.E. Tarjan

- [1973] "Dividing a Graph Into Triconnected Components," *SIAM J. Computing*, 2, 135-158.

C.L. Hwang, F.A. Tillman and M.H. Lee

- [1981] "System Reliability Evaluation Techniques for Complex/Large Systems--A Review," *IEEE Trans. Reliability*, R-30, 416-423.

Rubin Johnson

- [1982] "Some Combinatorial Aspects of Network Reliability," Ph.D. Thesis, Dept. of IEOR, University of California, Berkeley.

H. Kumamoto, K. Tanaka and K. Inoue

- [1977] "Efficient Evaluation of System Reliability by Monte Carlo Method," *IEEE Trans. Reliability* R-26, 311-315.

H. Kumamoto, K. Tanaka, K. Inoue and E.J. H

- [1980] "Dagger-sampling Monte Carlo for System Unavailability Evaluation," *IEEE Trans. Reliability*, R-29, 122-125.

David Lichtenstein

- [1981] "Dividing a Graph into Its 4-Connected Components," extended abstract, Dept. of Computer Science, Yale University, New Haven, Connecticut.

P.M. Lin, B.J. Leon and T.C. Huang

- [1976] "A New Algorithm for Symbolic System Reliability Analysis," *IEEE Trans. Reliability*, R-25, 2-15.

K.B. Misra

- [1970] "An Algorithm for the Reliability Evaluation of Redundant Networks," *IEEE Trans. Reliability*, 19, 146-151.

Fred Moskowitz

- [1958] "The Analysis of Redundancy Networks," *AIEE Trans. (Commun. Electron.)*, 77, 627-632.

Arnon Rosenthal

- [1974] "Computing Reliability of Complex Systems," Ph.D. Thesis, University of California, Berkeley.

A. Satyanarayana

- [1982] "A Unified Formula for Analysis of Some Network Reliability Problems," *IEEE Trans. Reliability*, R-31, 23-32.

A. Satyanarayana and M.K. Chang

- [1981] "Network Reliability and the Factoring Theorem," *Networks*, to appear. Also, ORC 81-12, Operations Research Center, University of California, Berkeley.

A. Satyanarayana and A. Prabhakar

- [1978] "New Topological Formula and Rapid Algorithm for Reliability Analysis of Complex Systems," *IEEE Trans. Reliability*, R-27, 82-100.

J. Sharma

- [1976] "Algorithm for Reliability Evaluation of a Reducible Network," *IEEE Trans. Reliability*, 25, 337-339.

R. Endre Tarjan

- [1972] "Depth-first Search and Linear Graph Algorithms," *SIAM J. Computing*, 1, 146-160.

W.T. Tutte

- [1966] *Connectivity in Graphs*, University of Toronto Press, Toronto.

J. Valdes, R.E. Tarjan, and E.L. Lawler

- [1981] "The Recognition of Series Parallel Digraphs," *SIAM J. Computing*, to appear.

Leslie G. Valiant

- [1979] "The Complexity of Enumeration and Reliability Problems," *SIAM J. Computing*, 8, 410-421.

R. Kevin Wood

- [1980] "Efficient Calculation of the Reliability of Lifeline Networks Subject to Seismic Risk," ORC 80-13, Operations Research Center, University of California, Berkeley.

MEI
83